UNIVERSITAT POLITÈCNICA DE CATALUNYA

Games in Machine Learning: Differentiable n-Player Games and Structured Planning

Carles Domingo Enrich

Treball Final de Grau
Grau en Matemàtiques (Facultat de Matemàtiques i Estadística)
Grau en Enginyeria Física (Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona)
Centre de Formació Interdisciplinària Superior

Work done at the Courant Institute of Mathematical Sciences
(New York University)

Supervisor: Joan Bruna Estrach (NYU)
Co-supervisor: Antonio Pascual Iserte (UPC)

May 2019

Abstract

This final project touches on two topics in machine learning that are somehow related to games, but that are actually loosely related to each other.

The first part of the thesis, which is the second one chronologically, is composed by Chapters 1, 2, 3 and Appendices A, B. Chapter 1 is a review a concepts and algorithms for differentiable n-player games. Chapter 2 and Appendix A consist of original work and show local convergence results for some of the algorithms discussed in Chapter 1. Chapter 3 (and Appendix B) contain a preliminary version of the article that we will submit to NeurIPS 2019 in May 2019. We provide global convergence guarantees (under certain assumptions) for a variant of the extra-gradient method that introduces stochasticity on the players. The article will also contain an experimental part, but is not included in this thesis. My coauthors for the work in Chapter 3 are Samy Jelassi, Arthur Mensch, Damien Scieur and Joan Bruna.

The second part of the thesis is composed by Chapter 4 and Appendix C. It is original work as well. Two planning methods that leverage the arithmetic structure of the environments are presented, together with experimental results. My coauthors for this part are David Folqué, Sainbayar Sukhbaatar, Arthur Szlam and Joan Bruna. This work was submitted unsuccessfully to ICML 2019 in January 2019.

Keywords: n-player differentiable games, Nash equilibria, extra-gradient method, planning in structured environments, reinforcement learning.

Acknowledgements

Vull agrair a en Joan Bruna que m'hagi ofert l'oportunitat de fer aquesta estada a NYU i que m'hagi dirigit en la recerca que he dut a terme.

Vull donar les gràcies al CFIS per haver proporcionat part del meu finançament.

I also want to thank my collaborators Samy, Arthur and Damien.

Per últim, vull donar les gràcies a la meva família.

Contents

Chapter 1

Differentiable n-Player Games

1.1 Framework

Definition 1.1. We define an **n-player differentiable game** to be the tuple $(n, x_1, ..., x_n, \ell_1, ..., \ell_n)$, where for all i such that $1 \leq i \leq n$, $x_i \subseteq \mathbb{R}^{d_i}$ and $\ell_i : X_1 \times \cdots \times X_n \to \mathbb{R}$ is twice differentiable.

We can express all the parameters of the game as

$$x = (x^{(1)}, x^{(2)}, ..., x^{(n)}) \in X := X_1 \times \cdots \times X_n \subseteq \mathbb{R}^d \tag{1.1}$$

Here, $d = \sum_{i=1}^{n} d_i$. The interpretation of the definition is that each player i has a loss function $\ell_i : X \to \mathbb{R}$ which has as input the vector of parameters x. As in any game theoretic setting, the player seeks to achieve the minimum value of ℓ_i possible but is only able to control $x^{(i)}$. We will write $x = (x^{(i)}, x^{(-i)})$, where $x^{(-i)}$ refers to all the parameters different from $x^{(i)}$.

Definition 1.2. We define the **simultaneous gradient**

$$F(x) = \begin{bmatrix} \nabla_1 \ell_1(x) \\ \nabla_2 \ell_2(x) \\ \vdots \\ \nabla_n \ell_n(x) \end{bmatrix} \tag{1.2}$$

We can write the Jacobian of the simultaneous gradient JF as

$$JF(x) = \begin{bmatrix} \nabla_{11} \ell_1(x) & \nabla_{12} \ell_1(x) & \cdots & \nabla_{1n} \ell_1(x) \\ \nabla_{21} \ell_2(x) & \nabla_{22} \ell_2(x) & \cdots & \nabla_{2n} \ell_2(x) \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_{n1} \ell_n(x) & \nabla_{n2} \ell_n(x) & \cdots & \nabla_{nn} \ell_n(x) \end{bmatrix} \tag{1.3}$$

The Jacobian of the simultaneous gradient is also commonly known as the Hessian of the game, although it is not the Hessian of any function in general.

Example 1.1. *Let $x^{(1)}, x^{(2)} \in \mathbb{R}^3$. We can define a two player game with losses:*

$$\ell_1(x^{(1)}, x^{(2)}) = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} \end{bmatrix} \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix}$$

$$\ell_2(x^{(1)}, x^{(2)}) = -\ell_1(x^{(1)}, x^{(2)}) = \begin{bmatrix} x_1^{(2)} & x_2^{(2)} & x_3^{(2)} \end{bmatrix} \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix}$$

This game is a bimatrix game. We can compute

$$\nabla_1 \ell_1 = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ x_3^{(2)} \end{bmatrix}$$

$$\nabla_2 \ell_2 = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{bmatrix}$$

$$JF = \begin{bmatrix} \nabla_{11}\ell_1 & \nabla_{12}\ell_1 \\ \nabla_{21}\ell_2 & \nabla_{22}\ell_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

1.1.1 Solution concepts

The most common solution concept in game theory is the Nash equilibrium. In our framework,

Definition 1.3. A **pure Nash equilibrium** is a vector $x_\star$ such that for all i and all $\tilde{x}^{(i)} \in X_i$, $\ell_i(x_\star^{(i)}, x_\star^{(-i)}) \leq \ell_i(\tilde{x}^{(i)}, x_\star^{(-i)})$.

A related concept is that of a local pure Nash equilibrium.

Definition 1.4. $x^\star$ is a **local pure Nash equilibrium** if for all i there exists a neighborhood $U_i \subseteq X_i$ of $x_\star^{(i)}$ such that for all $\tilde{x}^{(i)} \in X_i$, $\ell_i(x_\star^{(i)}, x_\star^{(-i)}) \leq \ell_i(\tilde{x}^{(i)}, x_\star^{(-i)})$.

Pure Nash equilibria are obviously local pure Nash equilibria. In an analogy with global and local minima, the advantage of local Nash equilibria is that they correspond to certain conditions on the first and second derivatives of the loss functions. Specifically,

Lemma 1.5. *The following hold for any n-player differentiable game:*

(a) $F(x_\star) = 0$ *is a necessary condition for* $x_\star$ *to be a local Nash equilibrium.*

(b) $\forall i$ *st* $1 \leq i \leq n$, $\nabla_{ii}\ell_i(x_\star) \succeq 0$ *is a necessary condition for* $x_\star$ *to be a local Nash equilibrium.*

(c) $F(x_\star) = 0$ *together with* $\forall i$ *st* $1 \leq i \leq n$, $\nabla_{11}\ell_1(x_\star) \succ 0$ *is a sufficient condition for* $x_\star$ *to be a local Nash equilibrium.*

Proof. It follows from sufficient and necessary conditions on the Hessian for local minima. □

Definition 1.6. $\hat{x}$ is a **fixed point** if $F(\hat{x}) = 0$.

Definition 1.7. A general matrix M is **positive semidefinite** if $x^\top Mx \geq 0$ for all $x \neq 0$.

Equivalently, M is positive semidefinite if and only if its symmetric part $\frac{M+M^\top}{2}$ is positive definite.

Definition 1.8. A **stable fixed point** $\hat{x}$ is a fixed point such that $H(\hat{x})$ is positive semidefinite.

Lemma 1.9. *If* $JF(\hat{x})$ *is positive semidefinite, then all the diagonal blocks* $\nabla_{ii}\ell_i(\hat{x})$ *of* H *are positive semidefinite.*

Proof. Let y be vector of dimension d_i (the same dimension that the square matrix $\nabla_{ii}\ell_i$). We want to see that $y^\top(\nabla_{ii}\ell_i)y$. We consider the vector y' of dimension d (the same dimension as JF), equal to y in the components corresponding to player i and 0 elsewhere. Then, we know that $0 \leq y'^\top JFy' = y^\top(\nabla_{ii}\ell_i)y$. □

We write $(JF)_0 = JF - \mathrm{diag}(JF)$, that is

$$(JF)_0(x) = \begin{bmatrix} 0 & \nabla_{12}\ell_1(x) & \cdots & \nabla_{1n}\ell_1(x) \\ \nabla_{21}\ell_2(x) & 0 & \cdots & \nabla_{2n}\ell_2(x) \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_{n1}\ell_n(x) & \nabla_{n2}\ell_n(x) & \cdots & 0 \end{bmatrix}$$

Lemma 1.10. *If* $(JF)_0 = -(JF)_0^\top$*, then all the diagonal blocks* $\nabla_{ii}\ell_i$ *of* JF *are positive semidefinite if and only if* JF *is positive semidefinite.*

Proof. We only need to prove from left to right. Let

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Then,

$$y^\top J\omega y = \begin{bmatrix} (y^{(1)})^\top & (y^{(2)})^\top & \cdots & (y^{(n)})^\top \end{bmatrix} \begin{bmatrix} \nabla_{11}\ell_1 & \nabla_{12}\ell_1 & \cdots & \nabla_{1n}\ell_1 \\ \nabla_{21}\ell_2 & \nabla_{22}\ell_2 & \cdots & \nabla_{2n}\ell_2 \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_{n1}\ell_n & \nabla_{n2}\ell_n & \cdots & \nabla_{nn}\ell_n \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$$= \begin{bmatrix} (y^{(1)})^\top & (y^{(2)})^\top & \cdots & (y^{(n)})^\top \end{bmatrix} \begin{bmatrix} \nabla_{11}\ell_1 & \nabla_{12}\ell_1 & \cdots & \nabla_{1n}\ell_1 \\ -(\nabla_{12}\ell_1)^\top & \nabla_{22}\ell_2 & \cdots & \nabla_{2n}\ell_2 \\ \vdots & \vdots & \ddots & \vdots \\ -(\nabla_{1n}\ell_1)^\top & -(\nabla_{2n}\ell_2)^\top & \cdots & \nabla_{nn}\ell_n \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$$= \sum_{i=1}^{n} (y^{(i)})^\top (\nabla_{ii}\ell_i) y^{(i)} + \sum_{i=1}^{n} \sum_{j<i} (y^{(i)})^\top (\nabla_{ij}\ell_i) y^{(j)} - (y^{(j)})^\top (\nabla_{ij}\ell_i)^\top y^{(i)}$$

$$= \sum_{i=1}^{n} (y^{(i)})^\top (\nabla_{ii}\ell_i) y^{(i)} + \sum_{i=1}^{n} \sum_{j<i} (y^{(i)})^\top (\nabla_{ij}\ell_i) y_j - ((\nabla_{ij}\ell_i) y^{(j)})^\top y^{(i)}$$

$$= \sum_{i=1}^{n} (y^{(i)})^\top (\nabla_{ii}\ell_i) y^{(i)} \quad (1.4)$$

From the last expression, we conclude that if $\nabla_{ii}\ell_i$ are positive definite for all i, JF is positive semidefinite. □

Lemma 1.11. *If $(JF)_0 = -(JF)_0^\top$ for all $x \in X$ (or at least at the fixed points), then local Nash equilibria are stable fixed points.*

Proof. By Lemma 1.5 and Lemma 1.10. □

1.1.2 Simultaneous gradient descent

Definition 1.12. Simultaneous gradient descent is defined as the dynamics given by

$$x_{\tau+1}^{(i)} = x_\tau^{(i)} - \alpha \nabla_i \ell_i (x_\tau^{(i)}) \quad (1.5)$$

Equivalently, $x_{\tau+1} = x_\tau - \alpha F(x_\tau)$.

Simultaneous gradient descent is the most basic dynamics in this framework and is not always guaranteed to converge locally to local Nash equilibria.

Continuous simultaneous gradient corresponds to the ODE $\frac{d}{dt}x = -F(x)$.

1.2 Types of n-player differentiable games

1.2.1 Potential games

Definition 1.13. A game is **potential**[1] if there exists a function $\phi : \mathbb{R}^d \to \mathbb{R}$ such that for all i, $\hat{x}^{(i)}$, $\tilde{x}^{(i)}$, $x^{(-i)}$,

$$\phi(\hat{x}^{(i)}, x^{(-i)}) - \phi(\tilde{x}^{(i)}, x^{(-i)}) = \ell_i(\hat{x}^{(i)}, x^{(-i)}) - \ell_i(\tilde{x}^{(i)}, x^{(-i)})$$

Monderer & Shapley [1] showed that

Theorem 1.14. *A game is potential if and only if for any point $x \in X$*

$$\nabla_{ij}\ell_i = \nabla_{ij}\ell_j = (\nabla_{ji}\ell_j)^\top$$

The implication from left to right is easy to see

$$\nabla_{ij}\ell_j = \nabla_{ij}\phi = (\nabla_{ji}\phi)^\top = (\nabla_{ji}\ell_i)^\top = \nabla_{ij}\ell_i$$

$\nabla_{ij}\ell_i = (\nabla_{ji}\ell_j)^\top$ is equivalent to $J\omega$ being symmetric.

For potential games, it is easy to see that

$$F(x) = \begin{bmatrix} \nabla_1\ell_1(x) \\ \nabla_2\ell_2(x) \\ \vdots \\ \nabla_n\ell_n(x) \end{bmatrix} = \begin{bmatrix} \nabla_1\phi(x) \\ \nabla_2\phi(x) \\ \vdots \\ \nabla_n\phi(x) \end{bmatrix} = \nabla\phi(x)$$

$$JF(x) = H\phi(x)$$

Hence, in this case simultaneous gradient descent is actually gradient descent on the potential function ϕ. If gradient descent converges to a local minimum $\hat{x}$ of ϕ, $H\phi(\hat{x}) \succeq 0$, which implies $JF(\hat{x}) \succeq 0$. Hence, $\hat{x}$ is a stable fixed point.

[1] Note: Our definition of potential game corresponds to exact potential game in [1].

1.2.2 Hamiltonian games

Definition 1.15. Hamiltonian games are those for which the Hessian of the game is anti-symmetric, that is $JF = A = \frac{JF - JF^\top}{2}$.

For Hamiltonian games, we can define a Hamiltonian function $\mathcal{H}(x) = \frac{1}{2}||F(x)||^2$.

We restate Theorem 3 from [2] regarding Hamiltonian games:

Theorem 1.16. *When the game is Hamiltonian, we have:*

(a) $\nabla\mathcal{H} = A^\top F$

(b) $\langle F, \nabla\mathcal{H}\rangle = 0$. *This implies that continuous simultaneous gradient dynamics preserves* $\mathcal{H}$.

(c) If $JF(x)$ *is invertible everywhere and* $\lim_{||x||\to\infty} \mathcal{H}(x) = \infty$, *gradient descent on* $\mathcal{H}$ *converges to a Nash equilibrium.*

Proof. (a) $\nabla\mathcal{H} = \nabla\left(\frac{1}{2}F^\top F\right) = H^\top F = A^\top F$

(b) $\langle F, \nabla\mathcal{H}\rangle = F^\top A^\top F$ and $F^\top A^\top F = (F^\top A^\top F)^\top = F^\top A F = -F^\top A^\top F$, which implies $F^\top A^\top F = 0$. Now, if x follows the dynamics given by $\frac{d}{dt}x = -F(x)$, then

$$\frac{d}{dt}\mathcal{H}(x(t)) = D\mathcal{H}(x(t))\frac{d}{dt}x(t) = \langle\nabla\mathcal{H}(x(t)), \frac{d}{dt}x(t)\rangle = -\langle\nabla\mathcal{H}(x(t)), F(x(t))\rangle = 0$$

(c) Gradient descent on $\mathcal{H}$ will converge to a point $\hat{x}$ where $\nabla\mathcal{H}(\hat{x}) = 0$, hence $JF(\hat{x})^\top F(\hat{x}) = 0$. Since JF is invertible by assumption, $F(\hat{x}) = 0$. Since JF is antisymmetric, it is positive semidefinite and hence $\hat{x}$ is a stable fixed point.

□

1.2.3 Two player zero-sum games

Definition 1.17. Two player zero-sum games are two player games such that $\ell_1(x) + \ell_2(x) = 0$.

This implies that $\nabla_{12}\ell_1 = -\nabla_{12}\ell_2 = -(\nabla_{21}\ell_2)^\top$. Hence, the Hessian of two player zero-sum games is

$$J\omega = \begin{bmatrix} -\nabla_{11}\ell_2 & -(\nabla_{21}\ell_2)^\top \\ \nabla_{21}\ell_2 & \nabla_{22}\ell_2 \end{bmatrix}$$

Although they are somewhat related, there is no inclusion between two player Hamiltonian games and two player zero-sum games:

- The diagonal blocks of JF might not be zero in zero-sum games, but they must be zero in the case of Hamiltonian games. This follows from the fact that $\nabla_{ii}\ell_i$ must be symmetric and antisymmetric for Hamiltonian games.
- Hamiltonian games might not fulfill $\ell_1(x) + \ell_2(x) = 0$, as the losses might contain linear terms on x that affect the sum but do not show up in H.

Remark 1.18. The game presented in Example 1.1 is Hamiltonian and two player zero-sum.

Remark 1.19. Two player zero-sum games fulfill $(JF)_0 = -(JF)_0^\top$ and hence by Lemma 1.11 all Nash equilibria are stable fixed points.

1.2.4 Zero-sum polymatrix games

The obvious generalization of zero-sum games to n-players would be to require that $\sum_{i=1}^n \ell_i = 0$. However, this definition is not very interesting because any n-player game could be expressed as an $n+1$-player game for which $\ell_{n+1} = -\sum_{i=1}^n \ell_i$.

An alternative generalization of the zero-sum condition is the notion of zero-sum polymatrix games (studied in [3] in the case of finite strategy sets).

Definition 1.20. Zero-sum polymatrix games are games in which for all i such that $1 \leq i \leq n$,

$$\ell_i(x) = \sum_{j \neq i} \ell_{ij}(x^{(i)}, x^{(j)})$$

where $\ell_{ij}(x^{(i)}, x^{(j)}) = -\ell_{ji}(x^{(j)}, x^{(i)})$, for all $i, j, x^{(i)}, x^{(j)}$.

Zero-sum polymatrix games are the same as two player zero-sum games when $n = 2$.

Lemma 1.21. *Zero-sum polymatrix games fulfill $(JF)_0 = -(JF)_0^\top$ and hence by Lemma 1.11 all Nash equilibria are stable fixed points.*

Proof. We just need to see that for all x, i and $j \neq i$, $\nabla_{ij}\ell_i = -(\nabla_{ji}\ell_j)^\top$. We have $\nabla_{ij}\ell_i = \nabla_{ij}\ell_{ij} = -\nabla_{ij}\ell_{ji} = -\nabla_{ij}\ell_j = -(\nabla_{ji}\ell_j)^\top$. □

1.2.5 Convex Nash Equilibrium Problems

Definition 1.22. An n-player differentiable game is a **Convex Nash Equilibrium Problem (CNEP)** if the function $l(x) := \sum_{i=1}^n \ell_i(x)$ is convex and for all i such that $1 \leq i \leq n$,

- $\ell_i(x) = \ell_i(x^{(i)}, x^{(-i)})$ is a convex function with respect to $x^{(i)}$ when $x^{(-i)}$ is kept fixed

- $\ell_i(x) = \ell_i(x^{(i)}, x^{(-i)})$ is a concave function with respect to $x^{(-i)}$ when $x^{(i)}$ is kept fixed

CNEPs are interesting because they allow for global convergence results that will be stated in the following sections. They are closely related to the Variational Inequality Problem (VIP), which we define below.

Definition 1.23. Given E a Banach space, an operator $F : E \to E^\star$ is **monotone** if for all $x, y \in E$,

$$\langle F(x) - F(y), x - y \rangle \geq 0$$

Definition 1.24. A **weak solution of the variational inequality (VI)** corresponding to the operator $F : Q \subseteq E \to E^\star$ is a point $x_\star \in Q$ such that

$$\langle F(x), x - x_\star \rangle \geq 0 \tag{1.6}$$

for all $x \in Q$.

Definition 1.25. A **strong solution of the variational inequality (VI)** corresponding to the operator $F : Q \subseteq E \to E^\star$ is a point $x^\star \in Q$ such that

$$\langle F(x_\star), x - x_\star \rangle \geq 0 \tag{1.7}$$

for all $x \in Q$.

Remark 1.26. If F is monotone, it is immediate to see that any strong solution of the VI is a weak solution of the VI.

Lemma 1.27. *If F is continuous and Q is convex, any weak solution of the VI is a strong solution of the VI.*

Proof. Let $x_\star$ be a weak solution of the VI. Let w be an arbitrary point in Q and define $x = x_\star + t(w - x_\star)$, which belongs to Q by convexity.

Now, by the definition of weak solution,

$$\langle F(x), x - x_\star \rangle = \langle F(x_\star + t(w - x_\star), t(w - x_\star) \rangle \geq 0$$

Dividing by t and taking the limit $t \to 0$, we get

$$\langle F(x_\star), w - x_\star \rangle \geq 0$$

for any $w \in Q$. □

Definition 1.28. The dual gap of the VI is defined as $Err_{VI} : Q \to \mathbb{R}$

$$Err_{VI}(x) = \sup_{u \in Z} \langle F(u), z - u \rangle \tag{1.8}$$

Definition 1.29. A subset $X \subseteq \mathbb{R}^n$ is a solid if it is compact, convex and its interior is non-empty.

The following theorem is very similar to Proposition 3.1 from [4]:

Theorem 1.30. *Assume that $E = \mathbb{R}^d$ with a norm that makes it a Banach space. Suppose X is a solid and $F : Q \subseteq E \to E^\star$ is a monotone operator, with int $X \subseteq Q$. Then,*

i The set $X_\star$ of solutions to (1.6) is a non-empty compact subset of X.

ii The function $Err_{VI}(x)$ is a closed convex non-negative function on X, finite everywhere on int X, and equal to 0 exactly at $X_\star$.

We restate Proposition 3.2 from [4] in our terminology.

Theorem 1.31 (Proposition 3.2 from [4])**.** *Let $X_1, ..., X_n$ as in Equation (1.1) be solids. Assume that the losses ℓ_i are such that the game is a CNEP. Then,*

1. *The simultaneous gradient F is a monotone operator.*
2. *When the losses ℓ_i are continuous on X, the weak solutions of the VI associated to F and X are exactly the Nash equilibria of the problem.*

Theorem 1.31 means that in CNEPs, the problem of finding Nash equilibria is equivalent to finding weak solutions of the VI. And Theorem 1.30 provides evidence that finding weak solutions of the VI is a well-posed problem when the operator is monotone.

1.3 Algorithms

1.3.1 Symplectic Gradient Adjustment

Definition 1.32. The symplectic gradient adjustment (SGA) is given by

$$x_{\tau+1} = x_\tau - \alpha \left(F(x_\tau) + \lambda A(x_\tau)^\top F(x_\tau) \right) = x_\tau - \alpha \left(I + \lambda A(x_\tau)^\top \right) F(x_\tau)$$

where $A = \frac{JF - JF^\top}{2}$.

The algorithm is presented in *The Mechanics of n-Player Differentiable Games* [2]. The authors propose to choose λ such that $\text{sign}(\lambda) = \text{sign}\left(\langle F, \nabla\mathcal{H}\rangle\langle A^\top F, \nabla\mathcal{H}\rangle\right)$, which means that it needs to be recomputed at each step. This choice makes SGA fulfill five desiderata that they consider reasonable to expect from dynamics that seek to find stable fixed points (although they do not imply convergence).

In the following we will consider a variation of SGA in which we set $\lambda = \alpha$. Hence, the SGA dynamics in this article will be

$$x_{\tau+1} = x_\tau - \alpha\left(I + \alpha A\left(x_\tau\right)^\top\right) F\left(x_\tau\right) \tag{1.9}$$

Since we are not choosing the sign of λ according to the criterion stated in [2], we cannot claim that the five desiderata will be fulfilled. In particular, we cannot claim that the last two will hold.

1.3.2 LookAhead and LOLA

LookAhead is another update that was presented and named in *Stable Opponent Shaping in Differentiable Games* [5], although it had been mentioned in [6].

Definition 1.33. In compact form, the LookAhead update can be written as

$$x_{\tau+1} = x_\tau - \alpha\left(I - \alpha(JF)_0\left(x_\tau\right)\right) F\left(x_\tau\right) \tag{1.10}$$

where $(JF)_0$ has outer diagonal blocks equal to JF and is zero in the diagonal blocks. That is,

$$(JF)_0(x) = \begin{bmatrix} 0 & \nabla_{12}\ell_1(x) & \cdots & \nabla_{1n}\ell_1(x) \\ \nabla_{21}\ell_2(x) & 0 & \cdots & \nabla_{2n}\ell_2(x) \\ \vdots & \vdots & \ddots & \vdots \\ \nabla_{n1}\ell_n(x) & \nabla_{n2}\ell_n(x) & \cdots & 0 \end{bmatrix}$$

[5] gives an intuition for LookAhead. LookAhead is related to Learning with Opponent-Learning Awareness (LOLA) [7]. In LOLA, each player computes its update as the gradient of a modified loss function in which the players have already moved. The modified loss function is

$$\ell_i(x^{(1)} + \Delta x^{(1)}, ..., x^{(i)}, ..., x^{(n)} + \Delta x^{(n)}) \approx \ell_i(x^{(1)}, ..., x^{(i)}, ..., x^{(n)}) + \sum_{j\neq i}(\nabla_j \ell_i(x))^\top \Delta x^{(j)}$$

Computing the gradient of this expression with respect to $x^{(i)}$ results in

$$\nabla_i \ell_i + \sum_{j\neq i}(\nabla_{ji}\ell_i(x))^\top \Delta x^{(j)} + (\nabla_j \ell_i(x))^\top \nabla_i \Delta x^{(j)} \tag{1.11}$$

Now, we suppose that the other players move according to gradient descent on their losses, that is, $\Delta x^{(j)} = -\alpha \nabla_j \ell_j$. Hence, Equation (1.11) becomes

$$\nabla_i \ell_i(x) - \alpha \left(\sum_{j \neq i} (\nabla_{ji} \ell_i(x))^\top \nabla_j \ell_j(x) + (\nabla_{ji} \ell_j(x))^\top \nabla_j \ell_i(x) \right)$$

And hence the update of player i according to LOLA is

$$x_{\tau+1}^{(i)} = x_\tau^{(i)} - \alpha \left(\nabla_i \ell_i(x) - \alpha \left(\sum_{j \neq i} (\nabla_{ji} \ell_i(x))^\top \nabla_j \ell_j(x) + (\nabla_{ji} \ell_j(x))^\top \nabla_j \ell_i(x) \right) \right)$$

The update of player i according to LookAhead amounts to dropping the second term of the summation. That is,

$$x_{\tau+1}^{(i)} = x_\tau^{(i)} - \alpha \left(\nabla_i \ell_i(x) - \alpha \left(\sum_{j \neq i} (\nabla_{ji} \ell_i(x))^\top \nabla_j \ell_j(x) \right) \right) \tag{1.12}$$

Going back to Equation (1.11), we see that in LookAhead we overlook the dependency of $\Delta x^{(j)}$ on $x^{(i)}$. It is also easy to see that putting the update in Equation (1.12) in a compact form corresponds to Equation (1.10).

1.3.3 Extra-gradient and mirror-prox

The extra-gradient method was introduced by Korpelevich [8] as a method to solve variational inequalities. There have been several works studying it and its generalized version, mirror-prox. All the theoretical results for extra-gradient or mirror-prox are under the assumption that the operator is monotone, and convex Nash equilibrium problems fulfill that condition (Theorem 1.31).

Definition 1.34. Each step of the extra-gradient method consists of two substeps:

$$x_{\tau+1/2} = x_\tau - \alpha F(x_\tau) \tag{1.13}$$

$$x_{\tau+1} = x_\tau - \alpha F(x_{\tau+1/2}) \tag{1.14}$$

Lemma 1.35. *The linearized version of extra-gradient is:*

$$x_{\tau+1} = x_\tau - \alpha \left(I - \alpha JF\left(x_\tau\right) \right) F\left(x_\tau\right) \tag{1.15}$$

Proof. From Equations (1.13) and (1.14), we get

$$x_{\tau+1} = x_\tau - \alpha F(x_\tau - \alpha F(x_\tau)) = x_\tau - \alpha F(x_\tau) + \alpha^2 JF(x_\tau) F(x_\tau) + o(\alpha^2) \tag{1.16}$$

□

Linearized extra-gradient is noticeably similar to LookAhead. The only difference is that in LookAhead players do not extrapolate themselves, while in LookAhead they do.

The mirror-prox method is the formulation of the extra-gradient method for Banach spaces. [9] is a good reference for mirror-prox and for the formulation of gradient descent for Banach spaces, known as mirror descent. I am including a brief overview on these two methods, as mirror-prox will be relevant in Chapter (3).

There is no notion of gradient in a Banach space, only of differential. Given E a Banach space and $f : E \to \mathbb{R}$, we will write $\nabla f : E \to E^\star$ to denote the differential because it is costumary in the optimization literature. In the following, we will assume $E = \mathbb{R}^n$ but we do not make any assumption on the norm (other than the space is Banach). In the case of games F will now denote the concatenation of the differentials.

Definition 1.36. Given a compact convex set $\mathcal{X} \subseteq E$ and $\mathcal{D} \subseteq E$ such that $\mathcal{X} \subseteq \text{int } \mathcal{D}$, the mirror map is defined as a continuously differentiable and μ-strongly convex function $\Phi : \mathcal{D} \to \mathbb{R}$.

Definition 1.37. Given a mirror map Φ, the Bregman divergence $D_\Phi : \text{int } D \times \text{int } D \to \mathbb{R}$ is defined as

$$D_\Phi(x, y) := \Phi(x) - \Phi(y) - \langle \nabla\Phi(y), x - y \rangle \tag{1.17}$$

It is clear that $\nabla\Phi$ maps $\mathcal{D}$ to $E^\star$.

Definition 1.38. Mirror descent is given by the following update:

$$\nabla\Phi(y_{\tau+1}) = \nabla\Phi(x_\tau) - \alpha F(x_\tau) \tag{1.18}$$

$$x_{\tau+1} = \text{argmin}_{x \in \mathcal{X} \cap \mathcal{D}} \; D_\Phi(x, y_{\tau+1}) \tag{1.19}$$

We can use the differential of the Legendre transform $\Phi^\star$ to compute $y_{\tau+1}$, as Equation (1.18) can be rewritten as

$$y_{\tau+1} = \nabla\Phi^\star(\nabla\Phi(x_\tau) - \alpha F(x_\tau)) \tag{1.20}$$

[9] contains convergence results for mirror descent in the monotone operator case (CNEP by Theorem 1.31). It is important to remark that it is necessary to take decreasing stepsizes to achieve convergence even when the losses are smooth. If we take stepsizes decreasing as $1/\sqrt{t}$, $\text{Err}_{VI}(\bar{x}_t)$ (Definition 1.8) decreases as $O(1/\sqrt{t})$, where $\bar{x}_t = \frac{1}{t}\sum_{\tau=1}^{t} x_\tau$.

Definition 1.39. Mirror-prox is given by the following update:

$$\nabla\Phi(y_{\tau+1/2}) = \nabla\Phi(x_\tau) - \alpha F(x_\tau) \tag{1.21}$$
$$x_{\tau+1/2} = \operatorname{argmin}_{x\in\mathcal{X}\cap\mathcal{D}} D_\Phi(x, y_{\tau+1/2}) \tag{1.22}$$
$$\nabla\Phi(y_{\tau+1}) = \nabla\Phi(x_\tau) - \alpha F(x_{\tau+1/2}) \tag{1.23}$$
$$x_{\tau+1} = \operatorname{argmin}_{x\in\mathcal{X}\cap\mathcal{D}} D_\Phi(x, y_{\tau+1}) \tag{1.24}$$

[10] and [11] provide convergence rates for mirror-prox in the non-noisy and noisy cases respectively, and the explanation in the latter is clearer. [9] contains some results as well. The most important result for mirror prox is that when the losses are L-smooth (gradients $\nabla_i \ell_i$ are L-Lipschitz), the game is a CNEP and the stepsizes are constant, $\mathrm{Err}_{VI}(\bar{x}_t)$ decreases as $O(1/t)$, where $\bar{x}_t = \frac{1}{t}\sum_{\tau=1}^{t} x_\tau$. Chapter (3) contains detailed results on mirror-prox.

For the following lemma we use the prox-mapping notation from [11].

Lemma 1.40. *Let the prox-mapping $P_z(\xi)$ be defined as*

$$P_z(\xi) = \mathit{argmin}_{u\in\mathcal{X}\cap\mathcal{D}}\{\Phi(u) + \langle \xi - \nabla\Phi(z), u\rangle\} = \mathit{argmin}_{u\in\mathcal{X}\cap\mathcal{D}}\{D_\Phi(z,u) + \langle \xi, u\rangle\} \tag{1.25}$$

An equivalent formulation of mirror-descent is

$$x_{\tau+1} = P_{x_\tau}(\alpha F(x_\tau)) \tag{1.26}$$

An equivalent formulation of mirror-prox is

$$x_{\tau+1/2} = P_{x_\tau}(\alpha F(x_\tau)) \tag{1.27}$$
$$x_{\tau+1} = P_{x_\tau}(\alpha F(x_{\tau+1/2})) \tag{1.28}$$

Proof. For mirror descent,

$$\begin{aligned} x_{\tau+1} &= \operatorname{argmin}_{x\in\mathcal{X}\cap\mathcal{D}} D_\Phi(x, y_{\tau+1}) = \operatorname{argmin}_{x\in\mathcal{X}\cap\mathcal{D}} \Phi(x) - \langle \nabla\Phi(y_{\tau+1}), x\rangle \\ &= \operatorname{argmin}_{x\in\mathcal{X}\cap\mathcal{D}} \Phi(x) - \langle \nabla\Phi(x_\tau) - \alpha F(x_\tau), x\rangle = \operatorname{argmin}_{x\in\mathcal{X}\cap\mathcal{D}} \langle \alpha F(x_\tau), x\rangle + D_\Phi(x, x_\tau) \end{aligned} \tag{1.29}$$

The proof for mirror-prox is analogous. □

Remark 1.41. If we choose $\Phi = \frac{1}{2}\|\cdot\|_2^2$ for mirror-prox, we recover extra-gradient.

Chapter 2

Local Convergence to Stable Fixed Points

2.1 Preliminaries

2.1.1 Notation

Instead of x, in this chapter we will use $\mathbf{w}$ for the vector of concatenated parameters. We do that to avoid confusion, as x is used in lemmas.

Also, we use $H := JF$ to denote the Hessian of the game. Analogously, $H_0 = (JF)_0$. Since H and H_0 will be used a lot, this will shorten expressions. Whenever there is no ambiguity, we will also use H to denote H evaluated at a stable fixed point $\hat{\mathbf{w}}$.

2.1.2 Eigenvalues and eigenvectors of matrices dependent on a parameter

In order to analyze the convergence of the algorithms, we will make use of some results about the spectrum of matrices that depend on a parameter. There are several articles on the subject. We will use several theorems from [12], which are stated below (adapting the notation and the statements).

We consider a function $A : \mathbb{R} \to \mathbb{R}^{n \times n}$, where $\mathbb{R}^{n \times n}$ denotes the space of square $n \times n$ matrices. We write α for the parameter and $A(\alpha)$ for the matrix. In all the following theorems we consider each matrix element $a_{ij}(\alpha)$ of $A(\alpha)$ to be an analytic function of α. We use $\lambda(\alpha)$ to refer to eigenvalues of $A(\alpha)$.

Theorem 2.1 (Theorem 1 of [12]). *If at $\alpha = \alpha_0$ there exists an eigenvalue $\lambda(\alpha_0)$ which is simple (unrepeated), then there exists a neighborhood of α_0 within which the function $\lambda(\alpha)$ is regular (analytic and single valued).*

As in [12], we will say that a matrix $A \in \mathbb{R}^n$ is of simple structure if it is diagonalizable over $\mathbb{C}$. That is, we allow for repeated eigenvalues but all Jordan blocks must be 1-dimensional.

Theorem 2.2 (Theorem 5 of [12]). *Let $A(\alpha_0)$ be a matrix of simple structure. If λ_0 is a simple eigenvalue of $A(\alpha_0)$ and $x, y \in \mathbb{R}^n$ are the right and left eigenvalues corresponding to λ_0 respectively, then*

$$\left.\frac{d\lambda}{d\alpha}\right|_{\alpha=\alpha_0} = \frac{y^\top \frac{dA}{d\alpha}(\alpha_0)x}{y^\top x}$$

Theorem 2.3 (Theorem 6 of [12]). *If $A(\alpha_0)$ is a matrix of simple structure and $A^{(q)}(\alpha_0)$ is the first non-vanishing derivative of $A(\alpha)$ at $\alpha = \alpha_0$, then the n eigenvalues $\lambda(\alpha)$ of $A(\alpha)$ are differentiable at least q times at λ_0 and their first $q-1$ derivatives all vanish at λ_0.*

Theorem 2.4 (Theorem 7 of [12]). *With the assumptions of Theorem 2.3, let $\lambda(\alpha_0)$ be an eigenvalue of $A(\alpha_0)$ with multiplicity m and let the columns of the $n \times m$ matrices X_m, Y_m span the right eigenspace $\mathcal{X}_1$ and the left eigenspace $\mathcal{Y}_1$ respectively. If these matrices are chosen so that $X_m^\top Y_m = I$, then the m derivatives $\lambda^{(q)}(\alpha_0)$ (of the m eigenvalues that coincide at α_0) are the eigenvalues of the matrix $Y_m^\top A^{(q)}(\alpha_0) X_m$.*

2.1.3 Relation between positive definiteness and eigenvalues

Definition 2.5. We define a square matrix $A \in \mathbb{R}^{n\times n}$ to be positive definite if $\forall x \in \mathbb{R}^n$ such that $x \neq 0$, we have $x^\top Ax > 0$.

Definition 2.6. Analogously, a matrix A is positive semidefinite if $\forall x \in \mathbb{R}^n$, we have $x^\top Ax \geq 0$.

Equivalently, a general matrix is positive (semi)definite if and only if its symmetric part is positive (semi)definite.

Lemma 2.7. *Let $A \in \mathbb{R}^{n\times n}$ positive definite. Then all the eigenvalues of A have positive real part.*

Proof. Let $\mu + i\nu$ be an eigenvalue of A with eigenvector $x + iy$, such that $\mu, \nu \in \mathbb{R}$ and $x, y \in \mathbb{R}^n$. Then,

$$((A-\mu) - i\nu)(x+iy) = 0 \implies (A-\mu)x + \nu y + i((A-\mu)y - \nu x) = 0$$

This implies that $(A-\mu)x + \nu y = 0$ and $(A-\mu)y - \nu x = 0$. And now,

$$x^\top (A-\mu)x + y^\top (A-\mu)y = -\nu x^\top y + \nu y^\top x = 0$$

Hence,

$$\mu = \frac{x^\top Ax + y^\top Ay}{x^\top x + y^\top y}$$

which is clearly positive since A is positive definite. □

Lemma 2.8. *Let $A \in \mathbb{R}^{n\times n}$ positive semidefinite. Then all the eigenvalues of A have nonnegative real part.*

Proof. Analogous to Lemma 2.7. □

2.1.4 Spectral radius and convergence

We recall that the spectral radius of a square matrix $A \in \mathbb{R}^{n\times n}$ is defined as the largest absolute value of its eigenvalues (real or complex).

Theorem 2.9. *Let $A \in \mathbb{C}^{n\times n}$ with spectral radius $\rho(A)$. Then $\rho(A) < 1$ if and only if $\lim_{k\to\infty} A^k = 0$. On the other hand, if $\rho(A) > 1$, $\lim_{k\to\infty} ||A^k|| = \infty$.*

Proof. Proof in *https://en.wikipedia.org/wiki/Spectral_radius.* □

Lemma 2.10 is used in the proof of Theorem 2.11. The proof for both is in Appendix A.

Lemma 2.10. *If $A \in \mathbb{R}^{n\times n}$ is such that $\rho(A) < 1$, then there exists $C > 0$ such that for any x_0 the sequence $(x_\tau)_{k\in\mathbb{N}}$ in $\mathbb{R}^n$ defined as $x_{\tau+1} = Ax_\tau$ fulfills $||x_\tau||_2 \leq C||x_0||_2$, $\forall k \geq 0$.*

Theorem 2.11. *If $A \in \mathbb{R}^{n\times n}$ and $\rho(A) < 1$, the sequence $(\tilde{x}_\tau)_{k\in\mathbb{N}}$ taking values in $\mathbb{R}^n$ and defined as*

$$\tilde{x}_{\tau+1} = A\tilde{x}_\tau + f(\tilde{x}_\tau)$$

where $\lim_{x\to 0} \frac{f(x)}{|x|} = 0$, there is a neighborhood U of 0 such that $(\tilde{x}_\tau)_{k\in\mathbb{N}}$ converges to 0 when $\tilde{x}_0 \in U$.

2.2 Conditions for local convergence

SGA, LookAhead and linearized extra-gradient (Equations (1.9), (1.10) can be expressed as a single expression:

$$\mathbf{w}_{\tau+1} = \mathbf{w}_\tau - \alpha\mathcal{X}(\mathbf{w}_\tau, \alpha)F(\mathbf{w}_\tau) \tag{2.1}$$

where $\mathcal{X}(\mathbf{w}, \alpha) = I + \alpha A(\mathbf{w})^\top$ for SGA, $\mathcal{X}(\mathbf{w}, \alpha) = I - \alpha H_0(\mathbf{w})$ for LookAhead and $\mathcal{X}(\mathbf{w}, \alpha) = I - \alpha H(\mathbf{w})$ for linearized extra-gradient.

Definition 2.12. We say that an algorithm is locally convergent to a fixed point $\hat{\mathbf{w}}$ if there exists a neighborhood U of $\hat{\mathbf{w}}$ such that if the first iterate $\mathbf{w}_0$ belongs to U, w_τ converges to $\hat{\mathbf{w}}$.

Lemma 2.13. *Let $\hat{\mathbf{w}}$ be a fixed point of the game. Then, we can write*

$$\mathbf{w}_{\tau+1} - \hat{\mathbf{w}} = (I - \alpha\mathcal{X}(\hat{\mathbf{w}}, \alpha)H(\hat{\mathbf{w}}))(\mathbf{w}_\tau - \hat{\mathbf{w}}) + o(|\mathbf{w}_\tau - \hat{\mathbf{w}}|) \tag{2.2}$$

Let $\tilde{H}(\alpha) = \mathcal{X}(\hat{\mathbf{w}}, \alpha)H(\hat{\mathbf{w}})$. If $\rho(I - \alpha\tilde{H}(\alpha)) < 1$, $\mathbf{w}_\tau$ converges to $\hat{\mathbf{w}}$ if $\mathbf{w}_0$ is in a certain neighborhood of $\hat{\mathbf{w}}$, i.e., there is local convergence.

Proof. We can write

$$\mathbf{w}_{\tau+1} - \hat{\mathbf{w}} = \mathbf{w}_\tau - \hat{\mathbf{w}} - \alpha\mathcal{X}(\mathbf{w}_\tau, \alpha)F(\mathbf{w}_\tau)$$

And now, since F is zero at the fixed point $\hat{\mathbf{w}}$ by definition, we can write the Taylor expansions

$$F(\mathbf{w}_\tau) = DF(\hat{\mathbf{w}})(\mathbf{w}_\tau - \hat{\mathbf{w}}) + o(|\mathbf{w}_\tau - \hat{\mathbf{w}}|) = H(\hat{\mathbf{w}})(\mathbf{w}_\tau - \hat{\mathbf{w}}) + o(|\mathbf{w}_\tau - \hat{\mathbf{w}}|)$$

$$\mathcal{X}(\mathbf{w}_\tau, \alpha) = \mathcal{X}(\hat{\mathbf{w}}, \alpha) + \mathcal{O}(|\mathbf{w}_\tau - \hat{\mathbf{w}}|)$$

We are keeping α fixed in the last equation. Thus, we get

$$\mathbf{w}_{\tau+1} - \hat{\mathbf{w}} = \mathbf{w}_\tau - \hat{\mathbf{w}} - \alpha\mathcal{X}(\hat{\mathbf{w}}, \alpha)H(\hat{\mathbf{w}})(\mathbf{w}_\tau - \hat{\mathbf{w}}) + o(|\mathbf{w}_\tau - \hat{\mathbf{w}}|) \tag{2.3}$$

We cannot directly apply Theorem 2.9 to Equation (2.2), because the equation contains the term $o(|\mathbf{w}_\tau - \hat{\mathbf{w}}|)$. To deal with this, we use Theorem 2.11, which is proved in Appendix A.1. We apply Theorem 2.11 with $\tilde{x}_\tau = \mathbf{w}_\tau - \hat{\mathbf{w}}$ and $A = I - \alpha\tilde{H}(\alpha)$. We get that if $\rho(I - \alpha\tilde{H}(\alpha)) < 1$, □

Lemma 2.14. *Assume that $H(\hat{\mathbf{w}})$ diagonalizes over $\mathbb{C}$. Denote by $\lambda_j(\alpha)$ the eigenvalues of $\tilde{H}(\alpha)$, where j is between 1 and n. If*

$$Re(\lambda_j(0)) > 0 \tag{2.4}$$

for all j between 1 and n, there is local convergence to $\hat{\mathbf{w}}$.

Proof. Since $\tilde{H}(0) = \mathcal{X}(\hat{\mathbf{w}}, 0)H(\hat{\mathbf{w}}) = H(\hat{\mathbf{w}})$, we know $\tilde{H}(0)$ diagonalizes over $\mathbb{C}$ (which in Section 2.1.2 we refer to as having simple structure).

The eigenvalues of $I - \alpha\tilde{H}(\alpha)$ are $1 - \alpha\lambda_j(\alpha)$. Since $\rho(I - \alpha\tilde{H}(\alpha)) = \max_j |1 - \alpha\lambda_j(\alpha)|$, the condition on the spectral radius means that for all j

$$|1 - \alpha\lambda_j(\alpha)| < 1 \iff |1 - \alpha\lambda_j(\alpha)|^2 < 1 \iff (1 - \alpha\lambda_j^*(\alpha))(1 - \alpha\lambda_j(\alpha)) < 1 \tag{2.5}$$

Obviously, for $\alpha = 0$ both sides are equal. By Theorem 2.3 and by the fact that the elements of $\tilde{H}(\alpha)$ are analytic on α (they are in fact affine functions), all the eigenvalues $\lambda_j(\alpha)$ are differentiable functions of α. Hence, a sufficient condition for Equation (2.5) to hold for some $\alpha > 0$ is

$$\frac{d}{d\alpha}((1-\alpha\lambda_j^*(\alpha))(1-\alpha\lambda_j(\alpha)))\Big|_{\alpha=0} < 0$$

We develop this expression

$$\begin{aligned}\frac{d}{d\alpha}((1-\alpha\lambda_j^*(\alpha))(1-\alpha\lambda_j(\alpha)))\Big|_{\alpha=0} &= \frac{d}{d\alpha}(1-\alpha(\lambda_j(\alpha)+\lambda_j^\star(\alpha))+\alpha^2\lambda_j(\alpha)\lambda_j^\star(\alpha))\Big|_{\alpha=0} = \\ &- (\lambda_j(\alpha)+\lambda_j^\star(\alpha))\Big|_{\alpha=0} - \alpha\left(\frac{d}{d\alpha}\lambda_j(\alpha)+\frac{d}{d\alpha}\lambda_j^\star(\alpha)\right)\Big|_{\alpha=0} + 2\alpha\lambda_j(\alpha)\lambda_j^\star(\alpha)\Big|_{\alpha=0} \\ &+ \alpha^2\frac{d\lambda_j(\alpha)}{d\alpha}\lambda_j^\star(\alpha)\Big|_{\alpha=0} + \alpha^2\lambda_j(\alpha)\frac{d\lambda_j^\star(\alpha)}{d\alpha}\Big|_{\alpha=0} = -2\mathrm{Re}(\lambda_j(0)) < 0 \quad (2.6)\end{aligned}$$

□

A sufficient condition for $\mathrm{Re}(\lambda_j(0)) > 0$ for all j is H being positive definite, by Lemma 2.7. However, H being positive semidefinite is not enough to ensure positivity of the real part of its eigenvalues. As a counterexample, a positive semidefinite matrix can have 0 as an eigenvalue.

$\mathrm{Re}(\lambda_j(0)) < 0$ implies that there is no local convergence. We look more carefully at the case for which $\mathrm{Re}(\lambda_j(0)) = 0$.

Lemma 2.15. *Let $\hat{\mathbf{w}}$ be a stable fixed point. As in Theorem 2.14, assume that $H(\hat{\mathbf{w}})$ diagonalizes over $\mathbb{C}$ and let $\lambda_j(\alpha)$ be the eigenvalues of $\tilde{H}(\alpha)$, where j is between 1 and n. Assume that $Re(\lambda_j(0)) \geq 0$ for all j between 1 and n, and that if $Re(\lambda_j(0)) = 0$,*

$$\lambda_{j,0}^2 < 2Re\left(\frac{d}{d\alpha}\lambda_j(0)\right) \quad (2.7)$$

Then, there is local convergence to $\hat{\mathbf{w}}$.

Proof. The case $\mathrm{Re}(\lambda_j(0)) > 0$ for all j is covered by Lemma 2.14. When $\mathrm{Re}(\lambda_j(0)) = 0$, the second order sufficient condition to have $\|1-\alpha\lambda_j^*(\alpha)\|_2^2 < 1$ for $\alpha > 0$ small is:

$$\frac{d^2}{d\alpha^2}((1-\alpha\lambda_j^*(\alpha))(1-\alpha\lambda_j(\alpha)))\Big|_{\alpha=0} < 0 \quad (2.8)$$

We develop this expression keeping only the terms that might not evaluate to 0

$$\frac{d^2}{d\alpha^2}((1-\alpha\lambda_j^*(\alpha))(1-\alpha\lambda_j(\alpha)))\Big|_{\alpha=0} = -\left(\frac{d}{d\alpha}\lambda_j(\alpha)+\frac{d}{d\alpha}\lambda_j^\star(\alpha)\right)\Big|_{\alpha=0}$$
$$-\left(\frac{d}{d\alpha}\lambda_j(\alpha)+\frac{d}{d\alpha}\lambda_j^\star(\alpha)\right)\Big|_{\alpha=0}+2\lambda_j(\alpha)\lambda_j^\star(\alpha)\Big|_{\alpha=0} = -4\mathrm{Re}\left(\frac{d}{d\alpha}\lambda_j(\alpha)\Big|_{\alpha=0}\right)+2|\lambda(0)|^2<0 \tag{2.9}$$

Since $\lambda(0)$ is imaginary, we can write $\lambda(0) = i\lambda_{j,0}$. Then, we end up with

$$\lambda_{j,0}^2 < 2\mathrm{Re}\left(\frac{d}{d\alpha}\lambda_j(0)\right) \tag{2.10}$$

as the sufficient condition.

To be able to write Equation (2.8) we must check that $\lambda_j(\alpha)$ is twice differentiable. The regularity of λ_j is deduced from the discussion in [12] after the statement of Theorem 3 of that article. □

2.3 Local convergence for SGA

Theorem 2.16. *Let $(\mathbf{w}_\tau)_{k\in\mathbb{N}}$ be a sequence on $\mathbb{R}$ built according to the Symplectic Gradient Adjustment given by Equation (1.9). Let $\hat{\mathbf{w}}$ be a stable fixed point, that is, $F(\hat{\mathbf{w}}) = 0$ and $H(\hat{\mathbf{w}}) \succeq 0$. Let us assume also that $H(\hat{\mathbf{w}})$ is invertible and diagonalizable (over $\mathbb{C}$).*

Then, there exists a neighborhood $U \subseteq \mathbb{R}^d$ of $\hat{\mathbf{w}}$ such that if $\hat{\mathbf{w}}_0 \in U$, $\mathbf{w}_\tau$ converges to $\hat{\mathbf{w}}$, i.e., there is local convergence to $\hat{\mathbf{w}}$.

Proof. The proof consists on checking that for each eigenvalue $\lambda_j(\alpha)$ of $\tilde{H}(\alpha)$, either the first order condition (Equation (2.4)) holds, or $\mathrm{Re}(\lambda_j(0)) = 0$ and the second order condition (Equation (2.7)) holds.

Step 1. Since $H(\hat{\mathbf{w}}) = \tilde{H}(0)$ is positive semidefinite by hypothesis, by Lemma 2.8 all its eigenvalues have nonnegative real part. The assumptions for Lemma 2.14 are satisfied whenever all the eigenvalues have strictly positive real part and the proof ends here for this case. The rest of the proof will deal with the case in which $H(\hat{\mathbf{w}}) = \tilde{H}(0)$ has some imaginary eigenvalues.

Step 2. We use the notation $H = H(\hat{\mathbf{w}})$. Let us consider the matrix $H^2 + 2A^\top H$. We can rewrite it as

$$H^2 + 2A^\top H = H^2 + 2\frac{H^\top - H}{2}H = H^\top H$$

Since H is assumed to be invertible, we conclude that $H^2 + 2A^\top H$ is positive definite.

Now, $H + \alpha(H^2 + 2A^\top H)$ is also positive definite for $\alpha > 0$, since H is positive semidefinite. We define $\hat{H}(\alpha) = H + \alpha(H^2 + 2A^\top H)$.

Step 3 (simple imaginary eigenvalues). Let $i\lambda_{j,0}$ be a simple imaginary eigenvalue of H with right eigenvector x_j and left eigenvector y_j. Let $\hat{\lambda}_j(\alpha)$ be the eigenvalue of $\hat{H}(\alpha)$ such that $\hat{\lambda}_j(0) = i\lambda_{j,0}$, which is a regular function of α by Theorem 2.1.

By Theorem 2.2, $\hat{\lambda}_j(\alpha)$ satisfies

$$\frac{d}{d\alpha}\hat{\lambda}_j(0) = \frac{y_j^\top \frac{d\hat{H}}{d\alpha}(0)x_j}{y_j^\top x_j}$$

And $\frac{d\hat{H}(\alpha)}{d\alpha} = H^2 + 2A^\top H$, which means

$$\frac{d}{d\alpha}\hat{\lambda}_j(0) = \frac{y_j^\top H^2 x_j}{y_j^\top x_j} + \frac{y_j^\top (2A^\top H)x_j}{y_j^\top x_j} = (i\lambda_{j,0})^2\frac{y_j^\top x_j}{y_j^\top x_j} + 2\frac{y_j^\top A^\top H x_j}{y_j^\top x_j} = -\lambda_{j,0}^2 + 2\frac{y_j^\top A^\top H x_j}{y_j^\top x_j} \quad (2.11)$$

Since by definition $\tilde{H}(\alpha) = \mathcal{X}(\hat{\mathbf{w}}, \alpha)H(\hat{\mathbf{w}}) = H(\hat{\mathbf{w}}) + \alpha A(\hat{\mathbf{w}})^\top H(\hat{\mathbf{w}})$ for SGA, $\tilde{H}(0) = H(\hat{\mathbf{w}}) = \hat{H}(0)$. Hence, $i\lambda_{j,0}$ is also an eigenvalue of $\tilde{H}(0)$. As in Section 2.2, we use $\lambda_j(\alpha)$ to denote the eigenvalue of $\tilde{H}(\alpha)$ such that $\lambda_j(0) = i\lambda_{j,0}$. By Theorem 2.2,

$$\frac{d}{d\alpha}\lambda_j(0) = \frac{y_j^\top \frac{d\tilde{H}}{d\alpha}(0)x_j}{y_j^\top x_j}$$

And $\frac{d\tilde{H}(\alpha)}{d\alpha} = A^\top H$, which means

$$\frac{d}{d\alpha}\lambda_j(0) = \frac{y_j^\top A^\top H x_j}{y_j^\top x_j} \quad (2.12)$$

Combining Equations (2.11) and (2.12), we get

$$\frac{d}{d\alpha}\hat{\lambda}_j(0) = -\lambda_{j,0}^2 + 2\frac{d}{d\alpha}\lambda_j(0) \quad (2.13)$$

Step 3 (imaginary eigenvalues with multiplicity). Let $i\lambda_{j_1,0} = i\lambda_{j_2,0} = \cdots = i\lambda_{j_m,0}$ be an imaginary eigenvalue of H with multiplicity m, where $j_1, ..., j_m$ are between 1 and n. We will write $J := j_1, ..., j_m$ and $i\lambda_{J,0} := i\lambda_{j_1,0} = \cdots = i\lambda_{j_m,0}$. Let X_J the $d \times m$ right eigenvalue matrix and Y_J the $d \times m$ left eigenvalue matrix. Choose X_J, Y_J such that $Y_J^T X_J = I_d$. Let $\hat{\lambda}_{j_1}(\alpha), ..., \hat{\lambda}_{j_m}(\alpha)$ be the eigenvalues of $\hat{H}(\alpha)$ such that $\hat{\lambda}_{j_1}(0) = \cdots = \hat{\lambda}_{j_m}(0) = i\lambda_{J,0}$. And let $\lambda_{j_1}(\alpha), ..., \lambda_{j_m}(\alpha)$ be the eigenvalues of $\tilde{H}(\alpha)$ such that $\lambda_{j_1}(0) = \cdots = \lambda_{j_m}(0) = i\lambda_{J,0}$.

Let $\text{eig}(A)$ denote the vector of eigenvalues of matrix A. Let us denote by $\frac{d}{d\alpha}\hat{\Lambda}_J(0)$ the vector formed by the m derivatives $\frac{d}{d\alpha}\hat{\lambda}_{j_1}(\alpha), ..., \frac{d}{d\alpha}\hat{\lambda}_{j_m}(\alpha)$. By Theorem 2.4,

$$\frac{d}{d\alpha}\hat{\Lambda}_J(0) = \text{eig}\left(Y_J^T \frac{d}{d\alpha}\hat{H}(0) X_J\right) \tag{2.14}$$

Let us define $\frac{d}{d\alpha}\Lambda_J(0)$ as the vector formed by the m derivatives $\frac{d}{d\alpha}\lambda_{j_1}(\alpha), ..., \frac{d}{d\alpha}\lambda_{j_m}(\alpha)$. Using the same reasoning as in Equations (2.11), (2.12) and (2.13), we get

$$\begin{aligned}\frac{d}{d\alpha}\hat{\Lambda}_J(0) = \text{eig}\left(-\lambda_{J,0}^2 I_d + 2Y_J^T A^\top H X_J\right) &= -\lambda_{j,0}^2 \mathbf{1}_d + 2\text{eig}\left(Y_J^T A^\top H X_J\right) \\ &= -\lambda_{J,0}^2 \mathbf{1}_d + 2\frac{d}{d\alpha}\Lambda_J(0)\end{aligned} \tag{2.15}$$

Hence, Equation (2.14) holds for all $\lambda_{j_1}(\alpha), ..., \lambda_{j_n}(\alpha)$ as well. That means that we can treat the multiplicity case and the simple case together for the rest of the proof.

Step 4. Since $\hat{H}(\alpha)$ is positive definite for $\alpha > 0$, we know that $\text{Re}(\hat{\lambda}_j(\alpha)) > 0$. $\text{Re}(\hat{\lambda}_j(0)) = \frac{1}{2}(i\lambda_{j,0} - i\lambda_{j,0}) = 0$ and that means that we must have

$$\begin{aligned}0 \leq \frac{d}{d\alpha}\text{Re}(\hat{\lambda}_j)(\alpha)\Big|_{\alpha=0} &= \frac{d}{d\alpha}\left(\frac{1}{2}\hat{\lambda}_j(\alpha) + \frac{1}{2}\hat{\lambda}_j^*(\alpha)\right)\Big|_{\alpha=0} \\ &= \frac{1}{2}\frac{d}{d\alpha}\hat{\lambda}_j(\alpha)\Big|_{\alpha=0} + \frac{1}{2}\left(\frac{d}{d\alpha}\hat{\lambda}_j(\alpha)\Big|_{\alpha=0}\right)^* = \text{Re}\left(\frac{d}{d\alpha}\hat{\lambda}_j(\alpha)\Big|_{\alpha=0}\right)\end{aligned} \tag{2.16}$$

Using Step 3, we get

$$0 \leq \text{Re}\left(-\lambda_{j,0}^2 + 2\frac{d}{d\alpha}\lambda_j(0)\right) = -\lambda_{j,0}^2 + 2\text{Re}\left(\frac{d}{d\alpha}\lambda_j(0)\right) \tag{2.17}$$

This inequality is the second order condition expressed in Equation (2.7) except that it is not strict. Achieving strictness for the inequality of Equation (2.17) concludes the proof.

Step 5. In the following the objective is to prove strictness in Equation (2.17). $H^\top H = H^2 + 2A^\top H$ is a symmetric positive definite matrix. Hence, it has real positive eigenvalues.

Let λ_m be the smallest eigenvalue of $H^\top H$. Let us consider $\bar{H}(\alpha) = H + \alpha\lambda_m I$. The eigenvectors of $\bar{H}(\alpha)$ are the same as the eigenvectors of H. Let be $x_{j,R} + ix_{j,I}$ be a right eigenvalue of H corresponding to the imaginary eigenvalue $i\lambda_{j,0}$, with $x_{j,R}, x_{j,I} \in \mathbb{R}^n$ (if $i\lambda_{j,0}$ has multiplicity, we just pick any of the right eigenvalues).

We use $x_{j,R}(\alpha) + ix_{j,I}(\alpha)$ to denote any right eigenvector of $\hat{H}(\alpha)$ with eigenvalue $\hat{\lambda}_j(\alpha)$ such that $\hat{\lambda}_j(0) = i\lambda_{j,0}$.

We use $\bar{\lambda}_j(\alpha)$ to denote any eigenvalue of $\bar{H}(\alpha)$ such that $\bar{\lambda}_j(0) = i\lambda_{j,0}$. It is easy to see that $\bar{\lambda}_j(\alpha)$ admits the expression $\bar{\lambda}_j(\alpha) = i\lambda_{j,0} + \alpha\lambda_m$.

Using the same reasoning as in the proof of Lemma 2.7, we get:

$$0 = \mathrm{Re}(i\lambda_{j,0}) = \frac{x_{j,R}^\top H x_{j,R} + x_{j,I}^\top H x_{j,I}}{x_{j,R}^\top x_{j,R} + x_{j,I}^\top x_{j,I}} \tag{2.18}$$

$$\mathrm{Re}(\hat{\lambda}_j(\alpha)) = \frac{x_{j,R}(\alpha)^\top (H + \alpha(H^2 + 2A^\top H)) x_{j,R}(\alpha) + x_{j,I}(\alpha)^\top (H + \alpha(H^2 + 2A^\top H)) x_{j,I}(\alpha)}{x_{j,R}(\alpha)^\top x_{j,R}(\alpha) + x_{j,I}(\alpha)^\top x_{j,I}(\alpha)} \tag{2.19}$$

$$\mathrm{Re}(\bar{\lambda}_j(\alpha)) = \frac{x_{j,R}^\top (H + \alpha\lambda_m I) x_{j,R} + x_{j,I}^\top (H + \alpha\lambda_m I) x_{j,I}}{x_{j,R}^\top x_{j,R} + x_{j,I}^\top x_{j,I}} \tag{2.20}$$

Now, since $H^2 + 2A^\top H \succeq \lambda I$,

$$\begin{aligned} &\frac{x_{j,R}(\alpha)^\top (H^2 + 2A^\top H) x_{j,R}(\alpha) + x_{j,I}(\alpha)^\top (H^2 + 2A^\top H) x_{j,I}(\alpha)}{x_{j,R}(\alpha)^\top x_{j,R}(\alpha) + x_{j,I}(\alpha)^\top x_{j,I}(\alpha)} \\ &\geq \frac{\lambda_m x_{j,R}(\alpha)^\top x_{j,R}(\alpha) + \lambda_m x_{j,I}(\alpha)^\top x_{j,I}(\alpha)}{x_{j,R}(\alpha)^\top x_{j,R}(\alpha) + x_{j,I}(\alpha)^\top x_{j,I}(\alpha)} = \lambda_m = \frac{\lambda_m x_{j,R}^\top x_{j,R} + \lambda_m x_{j,I}^\top x_{j,I}}{x_{j,R}^\top x_{j,R} + x_{j,I}^\top x_{j,I}} \end{aligned} \tag{2.21}$$

Also, since H is positive semidefinite and using Equation (2.18),

$$\frac{x_{j,R}(\alpha)^\top H x_{j,R}(\alpha) + x_{j,I}(\alpha)^\top H x_{j,I}(\alpha)}{x_{j,R}(\alpha)^\top x_{j,R}(\alpha) + x_{j,I}(\alpha)^\top x_{j,I}(\alpha)} \geq 0 = \frac{x_{j,R}^\top H x_{j,R} + x_{j,I}^\top H x_{j,I}}{x_{j,R}^\top x_{j,R} + x_{j,I}^\top x_{j,I}} \tag{2.22}$$

Adding inequality (2.21) times α and inequality (2.22) and using Equations (2.19) and (2.20), we get

$$\mathrm{Re}(\hat{\lambda}_j(\alpha)) \geq \mathrm{Re}(\bar{\lambda}_j(\alpha))$$

for all $\alpha \geq 0$. And since $\mathrm{Re}(\bar{\lambda}_j(\alpha)) = \alpha\lambda_m > 0$, this concludes the proof. □

2.4 Local convergence for LookAhead in zero-sum games

Lemma 2.17. *If $H_0 = -H_0^\top$ (that is, $(JF)_0 = -(JF)_0^\top$), SGA and LookAhead are the same algorithm. In particular, for two player zero-sum games and zero-sum polymatrix games, SGA and LookAhead are the same algorithm.*

Proof. LookAhead corresponds to $\mathcal{X}(x, \alpha) = I - \alpha H_0(x)$ (Equation (2.1)). SGA corresponds to $\mathcal{X}(x, \alpha) = I + \alpha A^\top(x)$. If $H_0 = -H_0^\top$, we use that the diagonal blocks $\nabla_{ii} l_i$ are symmetric to get the following result:

$$A^\top = \frac{H^\top - H}{2} = \frac{H_0^\top - H_0}{2} = -H_0 \tag{2.23}$$

$H_0 = -H_0^\top$ is fulfilled for two player zero-sum games and zero-sum polymatrix games by Remark 1.19 and Lemma 1.21. □

Theorem 2.18. *If $H_0 = -H_0^\top$, LookAhead (and SGA) are locally convergent at all local Nash equilibria $\mathbf{w}_\star$ such that $H(\mathbf{w}_\star)$ is invertible and diagonalizable (over $\mathbb{C}$).*

Proof. With the appropriate assumptions, local convergence to stable fixed points for LookAhead in the case $H_0 = -H_0^\top$ is implied by the result for SGA in Theorem 2.16 and by Lemma 2.17. By Lemma 1.11, $H_0 = -H_0^\top$ implies that all local Nash equilibria are stable fixed points. This concludes the proof. □

2.5 Local convergence for linearized extra-gradient

Theorem 2.19. *Let $\hat{\mathbf{w}}$ be a stable fixed point and assume that $H(\hat{\mathbf{w}})$ diagonalizes over $\mathbb{C}$ (no invertibility requirement). Then, linearized extra-gradient (Lemma 1.35) is locally convergent to $\hat{\mathbf{w}}$.*

Proof. Since $H(\hat{\mathbf{w}}) \succeq 0$, $\mathrm{Re}(\lambda_j(0)) \geq 0$ for all j between 1 and n by Lemma 2.8. If $\mathrm{Re}(\lambda_j(0)) > 0$ for all j between 1 and n, local convergence follows from Lemma 2.14. If there exists j between 1 and n such that $\mathrm{Re}(\lambda_j(0)) = 0$ (and hence, $\lambda_j(0) = i\lambda_{j,0}$), we just need to see that Equation (2.7) is fulfilled. Then, we conclude by Lemma 2.15.

Linearized extra-gradient corresponds to $\mathcal{X}(\mathbf{w}, \alpha) = I - \alpha H(\mathbf{w})$ (Equation (2.1)). Hence,

$$\tilde{H}(\alpha) = \mathcal{X}(\hat{\mathbf{w}}, \alpha) H(\hat{\mathbf{w}}) = (I - \alpha H(\hat{\mathbf{w}})) H(\hat{\mathbf{w}}) = H(\hat{\mathbf{w}}) - \alpha H(\hat{\mathbf{w}})^2 \tag{2.24}$$

By Theorem 2.2, if $i\lambda_{j,0}$ is a simple eigenvalue

$$\frac{d}{d\alpha}\lambda_j(0) = \frac{y_j^\top \frac{d\tilde{H}}{d\alpha}(0) x_j}{y_j^\top x_j} = \frac{y_j^\top \frac{d\tilde{H}}{d\alpha}(0) x_j}{y_j^\top x_j} = \frac{y_j^\top (-H(\hat{\mathbf{w}})^2) x_j}{y_j^\top x_j} = \lambda_{j,0}^2$$

If $i\lambda_{J,0}$ has multiplicity, using the reasoning from Equation (2.14):

$$\frac{d}{d\alpha}\Lambda_J(0) = \mathrm{eig}\left(Y_J^T \frac{d}{d\alpha}\tilde{H}(0) X_J\right) = \mathrm{eig}\left(Y_J^T (-H(\hat{\mathbf{w}})^2) X_J\right) = \lambda_{J,0}^2 \mathbf{1}_d \tag{2.25}$$

Hence, for all j between 1 and n,

$$\lambda_{j,0}^2 < 2\lambda_{j,0}^2 = 2\mathrm{Re}\left(\frac{d}{d\alpha}\lambda_j(0)\right) \tag{2.26}$$

which is Equation (2.7). □

Chapter 3

Stochastic Opponent Extrapolation

Data-driven model training is increasingly relying on finding Nash equilibria with provable techniques, e.g., for GANs and multi-agent RL. In this paper, we analyse a new extragradient method, that performs gradient extrapolations and updates on a random subset of players at each iteration. This approach provably exhibits the same rate of convergence as full extragradient in non-smooth stochastic monotonous games. We propose an additional variance reduction mechanism for this to hold for smooth monotonous games. By reducing the memory footprint, our approach makes extrapolation amenable to massive multiplayer settings, and brings surprising empirical speed-ups. We demonstrate the efficiency of player sampling on large-scale non-smooth and non-strictly monotonous games. We show that the joint use of extrapolation and player sampling allows to train and find better generative models on CIFAR10.

3.1 Introduction

A growing number of recent models require dealing with multiple interacting objectives, such as generative adversarial networks [13], imaginative agents [14], hierarchical reinforcement learning [15, 16] and multi-agent reinforcement learning in general [17]. All these examples can be cast as games where players are modules that compete or cooperate.

Multi-agent problems are challenging for several reasons. First, there is no clear notion of objective to optimize since there are multiple ones. Many options are then possible: optimizing all the objectives (either simultaneously or alternatively) or generating more appropriated sequences of objectives to optimize [18–20]. Another more natural approach is the notion of Nash equilibrium in multi-agent systems. It consists in a solution where no player can do better by unilaterally changing its strategy. However, finding a mixed Nash equilibrium is known to be PPAD-complete [21]. Finally, a common practice to optimize single objective functions in

machine learning consists in running the gradient descent algorithm which is known to converge to local minima under mild conditions [22, 23]. However, since each loss depends on the other agents' parameters in multi-agent games, gradient descent fails at finding the Nash equilibrium even in very simple settings – see examples in [5, 24].

Despite all these drawbacks, it is possible to find one Nash equilibrium for some class of games. Among the numerous examples, we consider in this paper the class of *convex n-player games* (see below for a formal definition) for which the Nash equilibrium exists but may not be unique [25]. Finding a Nash equilibrium in this setting is equivalent to solve a variational inequality (VI) [25, 26]. Two modifications of standard gradient descent are known to solve the VIP problem and therefore to converge to a Nash equilibrium, *averaging* [27, 28] and *extrapolation* with averaging (commonly known as the *extragradient method* [8]). It is shown to be theoretically and empirically faster than averaging and has been extensively studied over the last years. The extrapolation can be understood as an *opponent shaping* step in a n-player game: the player looks one step in the future and anticipates the next moves of his opponents.

However, in games, extragradient requires a full pass over all the players and evaluating their full gradients in two different points. This operation may represent a bottleneck in terms of time and memory in large-scale multi-agent systems such as multitask learning where each task can be seen as a player [29–31], generative multi-adversarial networks [32] or consensus in computer science [33].

3.1.1 Organisation and contributions.

This paper analyzes a class of stochastic variants of the extragradient algorithm that aim at reducing the memory cost and computational complexity in convex n-player games. Since the main bottleneck is the gradient computation in large-scale games, we present a doubly-stochastic version of extragradient that involves two different samplings.

In our setting, we assume that for each player, the loss function is defined as an empirical mean over data estimated functions (as it is standard in machine learning). The doubly-stochastic updates consist in simultaneously using two different sampling strategies. First, analogously to stochastic gradient descent [22, 34], the gradients are computed by randomly selecting mini-batches of data-estimated gradients. Secondly, instead of computing at each step all the players gradients to make the extrapolation, we randomly select a mini-batch of players gradients. Note that this mini-batch is common for all the players and in that sense, the agents updates are made in a centralized manner.

We make a sharp analysis of the rate of convergence under minimal assumptions (convex game and non-smooth losses). In particular, we show that if the stepsize is modified accordingly to

the sampling strategy, sampling over players and data has no downside compared to full-batch extragradient algorithm. The constants in the rates may even be improved when the algorithm uses importance sampling. We also introduce a variance-reduced version of the algorithm to obtain faster rates in the case of smooth losses. Table 3.1 summarize the convergence rates of those methods.

We performed several numerical experiments that compare algorithms for n-players game (e.g. GANs or Particle Swarm).

TABLE 3.1: Convergence rates with respect to number of computations c, in existing work and in this paper. G is a bound on the gradient norm L the loss Lipschitz constant when available, σ^2 bounds the noise in gradient estimation, k is the number of sampled players among n, Ω is the diameter of the parameter space. See text for details.

Algorithm	Convergence rate (non-smooth case)	Convergence rate (smooth case)
Juditsky et al. [11]	$O\left(\sqrt{\frac{\Omega n^2(4G^2+\sigma^2)}{c}}\right)$	$O\left(\frac{\Omega L n^{3/2}}{c} + \sqrt{\frac{\Omega n^2\sigma^2}{c}}\right)$
Doubly stochastic	No VR: $O\left(\sqrt{\frac{\Omega n(3G^2 n+(\sigma^2-G^2)k)}{c}}\right)$	With VR: $O\left(\frac{\Omega L n^2}{\sqrt{k}c} + \sqrt{\frac{\Omega nk\sigma^2}{c}}\right)$

3.2 Related work

An important number of works analyze convergence in convex n-players games, see for example [35, 36]. In this paper, we focus on finding the Nash equilibrium in convex n-player games that is equivalent to solving a Variational Inequality (VI) [25, 26]. One algorithm to solve such problem is the extra-gradient algorithm originally introduced by Korpelevich [8] and developed by Nemirovski [10] and Nesterov [37]. A number of recent works proposed stochastic [11, 38] or adaptive variants [39] of the extra-gradient for stochastic variational inequalities where the gradient is a noisy estimate. On the other hand, [40] consider multi-agent games on networks and analyze a stochastic variant of extra-gradient that consists in randomly updating some players. However, they consider exact gradient estimates while we work with noisy estimates of the gradients from the data. Moreover, their analysis holds for smooth losses and the benefits of sampling players are not clearly stated. In this work, we unify these two approaches and study the case where we sample over players' losses and estimate noisy versions of their gradients. Several papers proposed variance-reduced variants for solving stochastic variational inequalities [41–43]. However, the variance reduction is applied to the noise arising from the data while we do it on the noise from the sampling of players.

On the other hand, a number of algorithms have been proposed in the non-convex setting under restricted assumptions on the game, for example WoLF in two-player two-action games [44], policy prediction in two-player two-action bimatrix games [6], AWESOME in repeated games [45],

Optimistic Mirror Descent in two-player bilinear zero-sum games [46] and Consensus Optimisation in two-player zero-sum games [47]. Mertikopoulos et al. [48] proved asymptotic convergence results for extragradient in a slightly non-convex setting. Other methods have also been proposed for GANs training including [24, 49, 50]. Globally, solving the non-convex variationnal inequality still remains an open problem.

Extra-gradient can also be seen as an *opponent shaping* method: in the extrapolation step, the player looks one step in the future and anticipates the next moves of his opponents. Many recent papers designed algorithms that make use of the opponents' information to converge to an equilibrium [5–7]. In particular, the Learning with Opponent-Learning Awareness (LOLA) algorithm is known for encouraging cooperation in cooperative games [7]. We show that it can be seen as mirror descent on a extrapolated loss. This is a different form of extrapolation from the one of extragradient since the extrapolation is made at the level of the loss.

Lastly, some recent works proposed algorithms to modify the dynamics of simultaneous gradient by adding an adjustment term in order to converge to the Nash equilibrium and avoid oscillations [2, 47, 51]. One caveat of these works is that they need to estimate the Jacobian which may be expensive in large-scale systems or even impossible when dealing with non-smooth losses as we consider in our setting. This line of research is orthogonal to this paper where we do not modify the dynamics and we just investigate the effect of the noise from the estimates of the gradients and from the sampling over players in extra-gradient.

3.3 Solving convex games with partial information

Before presenting the doubly-stochastic extra-gradient algorithm and its analysis in Section 3.4, we first present the general framework of games and the assumptions we make.

3.3.1 Convex n-player games and the variational inequality problem

In this section, we consider the problem of multi-agent learning as a n-player game where n players interact and each of them is interested in minimizing its loss. Among the numerous learning models that fit in this framework, the generative adversarial network (GAN) training can be cast as a game between two players called generator and discriminator. We provide a more formal definition of a game below.

Definition 3.1. A n-player game is given by a set of n players with parameters $\theta = (\theta^1, \dots, \theta^n) \in \Theta \subset \mathbb{R}^d$ where for each i, θ^i is the parameter of player i. Each player has a loss $\ell_i \colon \Theta \to \mathbb{R}$.

We assume that the parameter space Θ is compact, convex and non-empty. Therefore, it has a diameter Ω defined as $\Omega = \max_{u,z\in\Theta} \|u - z\|_2$. Additionally, we make two other assumptions on the structure of the game. First, regarding the players losses, we assume that they are convex.

Assumption 3.2. A n-player game is convex if each player's loss $\ell_i(\theta^i, \theta^{-i})$ is convex in its parameter θ^i and concave in θ^{-i}, where θ^{-i} contains all other players' parameters. Moreover, $\sum_{i=1}^n \ell_i(\theta)$ is convex in θ.

Secondly, we assume that for each i, each player's parameter θ^i belongs to a compact convex set Θ_i that is independent of the others. More formally, it amounts to decomposing the parameter space Θ as a Cartesian product.

Assumption 3.3. The parameters space Θ admits a Cartesian product decomposition i.e. $\Theta = \Theta_1 \times \cdots \times \Theta_n$ where $\theta^i \in \Theta_i \subset \mathbb{R}^{d_i}$ and $\sum_{i=1}^n d_i = d$.

One example of convex game that satisfies Assumptions 3.2 and 3.3 is rock-paper-scissors. The players losses are $\ell_1(x, y) = -\ell_2(x, y) = x^\top A y$ where A is the payoff matrix and x and y are the strategies of player 1 and 2. Remark that the losses satisfy Assumption 3.2. Moreover, the parameters x and y belong to the probability simplex in $\mathbb{R}^3$.

As it is common in machine learning, we assume that the losses are estimated from data and can be written as an empirical mean

$$\ell_i = \frac{1}{m} \sum_{j=1}^{m} \phi_i(\cdot, \xi_j), \tag{3.1}$$

where $\phi_i(\cdot, \xi_j)$ represents the loss associated to the sample j for player i. This sum is an empirical approximation of the population risk $\mathbb{E}[\ell(\cdot, \xi)]$ where ξ is a random variable encoding the randomness of the data. For example, in the case of GANs training, the losses of the discriminator and the generator take the form of (3.1) since we train both architectures on finite datasets of images.

In what follows, we write the derivative of loss k with respect to variable θ_i as $\nabla_i \ell_k = \nabla_{\theta^i} \ell_k$ for any i, k. We define the *simultaneous (sub)gradient* of the game to be

$$F := (\nabla_1 \ell_1, \ldots, \nabla_n \ell_n)^\top \in \mathbb{R}^d. \tag{3.2}$$

It corresponds to the concatenation of the (sub)gradients of each player's loss with respect to its parameter.

While the standard solution concept when we optimize a single objective function (a game with $n = 1$) is the minimum, one natural notion of equilibrium for arbitrary n corresponds to the *Nash equilibrium*.

Definition 3.4. A point $\theta_\star \in \mathbb{R}^d$ is a Nash equilibrium if, for all i, $\ell_i(\theta^i, \theta_\star^{-i}) \geq \ell_i(\theta_\star)$ for all $\theta^i \in \Theta_i$. Intuitively, the Nash equilibrium is a point where no player can benefit by changing his strategy while the other players keep theirs unchanged.

To our knowledge, little is known about the existence of (pure) Nash equilibria for non-convex games (see [50] and references therein for some results). Fortunately, finding a Nash equilibrium in a convex n-player game is related to solving the variational inequality.

Definition 3.5. Given $\Theta \subseteq \mathbb{R}^d$ compact convex and $F : \Theta \to \mathbb{R}^d$, the *variational inequality* (VI) in strong form is:

$$\text{find} \quad \theta_\star \in \Theta \quad \text{such that} \quad F(\theta_\star)^\top(\theta - \theta_\star) \geq 0 \quad \forall \theta \in \Theta. \tag{3.3}$$

The set of solutions to the VI is non-empty and compact when the operator F is *monotone* [25].

Definition 3.6 (From [25])**.** An operator $F\colon \Theta \to \mathbb{R}^d$ is monotone if $\forall \theta, \theta' \in \Theta$, $\langle F(\theta) - F(\theta'), \theta - \theta' \rangle \geq 0$.

For convex n-player games (Assumption 3.2), the simultaneous (sub)gradient F is a monotone operator. Moreover, the set of solutions to the VI coincides with the set of Nash equilibria and solving the VI is therefore sufficient to find Nash equilibria [25, 26]. The intuition behind this result is that equation (3.3) corresponds to the first-order necessary optimality condition applied to the losses of players.

The quantity that is used in the literature to quantify the inaccuracy of a solution θ in problem (3.3) is the dual VI gap defined as $\mathrm{Err}_{\mathrm{VI}}(\theta) = \max_{u \in \Theta} \langle F(u), \theta - u \rangle$. However, in the specific case of convex games, the *functional Nash error* is the usual performance measure. It is defined as

$$\mathrm{Err}_N(\theta) = \sum_{i=1}^{n} \left[\ell_i(\theta) - \min_{z \in \Theta_i} \ell_i(z, \theta^{-i}) \right]. \tag{3.4}$$

Intuitively, this sum over the players quantifies the loss incurred by a players when deviating from his choice given that other players stick to their choices. In Section 3.4, we provide bounds for the Nash error measure. However, they also bound the dual VI gap (see Lemmas B.3 and B.4 and in Appendix).

The two standard methods known to solve (3.3) are gradient descent with averaging [28] and the extra-gradient method [8]. There are some results for the former under Assumption 3.2, but in the general case it is shown to fail even in very simple settings – see [5, 24] for examples. The cause of failure is that the players do not extrapolate the behavior of others. The latter is the basis for the methods of this paper. It has been extensively analyzed under several settings

(see Section 3.2). In particular, [10] provides convergence results in the batch case and [11] in the case of noisy gradients.

Extra-gradient consists in two steps: first, computing an extrapolated point and then using this point for performing a gradient step:

$$\begin{aligned} \text{Compute extrapolated point: } \theta_{\tau+1/2} &= P_\Theta[\theta_\tau - \gamma_\tau F(\theta_\tau)], \\ \text{Make a gradient step: } \theta_{\tau+1} &= P_\Theta[\theta_\tau - \gamma_\tau F(\theta_{\tau+1/2})], \end{aligned} \tag{3.5}$$

where $P_\Theta[\cdot]$ is the proximal mapping and corresponds to the projection onto the constraint set Θ. In large-scale systems, this algorithm may be computationally expensive due to the high number of gradient computations. To alleviate this issue, we present algorithms that involve fewer players gradients estimations.

3.3.2 Partial extrapolation in extra-gradient

In this subsection, we present an algorithm that performs doubly-stochastic updates (with and without variance-reduction) in extra-gradient for convex n-player games. We proceed to its analysis in Section 3.4.

As mentioned above, the main bottleneck of extra-gradient is the expensive computation of gradients for all the players. While standard extra-gradient requires a full pass on both players and data, we instead compute *doubly-stochastic simultaneous gradients*. This corresponds to a simultaneous gradient that is affected by two sources of noise. First, after sampling a mini-batch $\mathcal{P}$ of size k over players, only the gradients corresponding to this batch are computed. Secondly, after sampling a mini-batch $\mathcal{D}$ of size l over the data-estimated functions in the finite sum (3.1), the loss is estimated by summing up all the elements in this sample. More formally, the doubly-stochastic simultaneous gradient $\tilde{F}$ is

$$\tilde{F} := (\tilde{F}^{(1)}, \dots, \tilde{F}^{(n)})^\top \in \mathbb{R}^d \text{ where } \tilde{F}^{(i)} = \begin{cases} \frac{n}{k} \cdot \frac{1}{l} \sum_{j \in \mathcal{D}} \nabla_i \phi_i(\cdot; \xi_j) & \text{if } i \in \mathcal{P} \\ 0_{d_i} & \text{otherwise} \end{cases}, \text{ for } i \in [n], \tag{3.6}$$

Remark that we add a factor n/k in equation (3.6) to ensure that the doubly-stochastic simultaneous gradient is an unbiased estimator of the simultaneous gradient.

To obtain faster rates in convex games with smooth losses (precised below), we compute variance-reduced estimate of the simultaneous gradient. Variance reduction is a technique known to accelerate convergence under smoothness assumptions in similar settings. While [41–43] apply variance reduction on the noise from the gradient estimates, we apply it to the stochasticity

coming from the sampling over the players. Inspired by the SAGA algorithm [52], we implement this idea in Algorithm 1.

Algorithm 1 Variance reduced estimate of the simultaneous gradient with doubly-stochastic sampling

function ESTIMATE_GRADIENT_VR(θ, k, l, G)
  Input: point $\theta \in \mathbb{R}^d$, size of mini-batch over players $k \leq n$, size of mini-batch over data $l \leq m$, table of previous gradient estimates $G \in \mathbb{R}^d$.
  Sample mini-batches over players $\mathcal{P}$ and over data $\mathcal{D}$.
  Compute $\tilde{F}(\theta)$ as specified in equation (3.6).
  for $i \in \mathcal{P}$ **do**
    Compute $\bar{F}^{(i)} \leftarrow \tilde{F}^{(i)}(\theta) - (1 - \frac{k}{n})G^{(i)}$ and update $G^{(i)} \leftarrow \tilde{F}^{(i)}(\theta)$
  for $i \notin \mathcal{P}$ **do**
    Set $\bar{F}^{(i)} \leftarrow G^{(i)}$.
  Return the estimate $\bar{F} = (\bar{F}^{(1)}, ..., \bar{F}^{(n)})$, updated table G.

The doubly-stochastic extra-gradient algorithm is detailed in Algorithm 2. The main difference with extra-gradient (3.5) is that the gradients are estimated by the oracles either defined in equation (3.6) or in Algorithm 1. Indeed, we can recover standard extra-gradient by setting $k = n$ in these oracles. Remark that both sampling over players and data are uniformly performed.

Algorithm 2 Doubly stochastic extra-gradient.

1: **Input**: initial point $\theta_0 \in \mathbb{R}^d$, stepsizes $(\gamma_\tau)_{\tau\in[t]}$, mini-batch size over the players $k \in [n]$, mini-batch size over the data $l \in [m]$.
2: If variance-reduction, initialize $G \leftarrow \tilde{F}(\theta_0)$ as in Equation (3.6) with full batch of players.
3: **for** $\tau = 0, \dots, t$ **do**
4: Compute $\tilde{F}_{\tau+1/2}$ evaluating the gradient estimate with variance reduction (Algorithm 1) or without (equation (3.6)) at point θ_τ.
5: Extrapolation step: $\theta_{\tau+1/2} \leftarrow P_\Theta[\theta_\tau - \gamma_\tau \tilde{F}_{\tau+1/2}]$.
6: Compute $\tilde{F}_{\tau+1}$ evaluating the gradient estimate with variance reduction (Algorithm 1) or without (equation (3.6)) at point $\theta_{\tau+1/2}$.
7: Gradient step: $\theta_{\tau+1} \leftarrow P_\Theta[\theta_\tau - \gamma_\tau \tilde{F}_{\tau+1}]$.
8: **Return** $\hat{\theta}_t = \left[\sum_{\tau=0}^t \gamma_\tau\right]^{-1} \sum_{\tau=0}^t \gamma_\tau \theta_\tau$.

Importance sampling. Different sampling strategies are possible in the computation of the doubly-stochastic simultaneous gradient (3.6) and its variance-reduced version in Algorithm 1. In particular, we consider importance sampling that consists in selecting the i-th player with a probability depending on G_i where $\|\nabla_i \ell_i\|_2 \leq G_i$. This has shown to improve the constants in the rates (Section 3.4) and to lead to faster convergence numerically in some cases.

3.4 Theoretical guarantees

We state our main convergence results in this section. As announced, we derive rates for the algorithms mentioned in subsection 3.3.2 following the analysis by [11]. In the appendices, the results are proven for a generalized version of extra-gradient named mirror-prox (Juditsky et al. [11])[1]. Nonetheless, in this section we stick to the Euclidean notation for simplicity. We separately consider the two following assumptions:

(**A**) Non-smooth assumption: For each $i \in [n]$, the loss ℓ_i has a bounded (sub)gradient, i.e. $\sup_{g\in\partial_i \ell_i(\theta)} \|g\|_2 \leq G_i$ for all $\theta \in \Theta$. In this case, we also define the quantity $G = \sqrt{\sum_{i=1}^n G_i^2/n}$.

(**B**) Smooth assumption: For each $i \in [n]$, the loss ℓ_i is once-differentiable and L-smooth, i.e. $\|\nabla_i \ell_i(\theta) - \nabla_i \ell_i(\theta')\|_2 \leq L\|\theta - \theta'\|$, for $\theta, \theta' \in \Theta$.

[1] Mirror descent and mirror-prox are the generalizations of gradient descent and extra-gradient when the space of parameters is only assumed to be Banach. The lack of a scalar product implies that (sub)gradients are not defined, only (sub)differentials. These methods make use of a mirror map (see [9]) to incorporate information from the sub(differential) into the optimization dynamics.

The appropriate comparison between bounds is not at a fixed number of iterations, but at a fixed number of total gradient computations.

Definition 3.7. We define $c(t)$ as the number of gradients estimates $\nabla_i \ell_i$ computed up to iteration t.

Since at each iteration of Algorithm 2 the gradient estimate is computed twice and it requires k gradient estimates each time, $c(t) = 2kt$. Hence, $t(c) = c/2k$. We put the bounds in terms of c in the statement of the theorems.

For stochastic extra-gradient (no player subsampling), the arguments from Juditsky et al. [11] yield the following rates.

Theorem 3.8 (From [11])**.** *If we consider a convex n-player game for which (**A**) holds, the rate of convergence achieved by stochastic extra-gradient (i.e., $k = n$ in Algorithm 2) is*[2]

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_{t(c)})\right] \leq \sqrt{\frac{8\Omega n^2}{c}\left(4G^2 + \sigma^2\right)} \quad \textit{when for all } \tau \in [t(c)],\ \gamma_\tau = \gamma = \sqrt{\frac{4\Omega}{(n\sigma^2 + 4nG^2)t}}. \tag{3.7}$$

*Assuming (**B**), the rate of convergence achieved by stochastic extra-gradient is*

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_{t(c)})\right] \leq \max\left\{\frac{7\Omega L n^{3/2}}{c}, 14n\sqrt{\frac{2\Omega\sigma^2}{3c}}\right\} \quad \textit{when } \gamma_\tau = \gamma = \min\left\{\frac{1}{\sqrt{3n}L}, \sqrt{\frac{2\Omega}{7t\sigma^2}}\right\}. \tag{3.8}$$

We state the convergence result for doubly-stochastic extra-gradient under the non-smooth assumption (**A**).

Theorem 3.9. *We consider a convex n-player game where Assumption (**A**) holds. Assume that Algorithm 2 is run without variance reduction. In terms of the number of gradient computations c, the rate of convergence in expectation is*

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_{t(c)})\right] \leq 4\sqrt{\frac{\Omega n}{c}\left(3nG^2 + k(\sigma^2 - G^2)\right)} \ \textit{when we set } \gamma_\tau = \gamma = \sqrt{\frac{2\Omega}{n\left(\frac{3nG^2}{k} + \sigma^2 - G^2\right)t}}. \tag{3.9}$$

We now provide the result for the case in which the losses are differentiable and smooth (**B**).

[2] This result is not explicitly stated in [11].

Theorem 3.10. *We consider a convex n-player game where Assumption (**B**) holds. Assume that we run Algorithm 2 with variance reduction. For all $\tau \in [t]$, set a constant stepsize*

$$\gamma_\tau = \gamma = \min\left\{\gamma_{\max}, \frac{1}{4}\sqrt{\frac{\Omega}{n\sigma^2 t}}\right\} \text{ with } \gamma_{\max} = \left\{\frac{k^{3/2}}{\sqrt{\left(1-\frac{k}{n}\right)\left(2-\frac{k}{n}\right)}}\frac{1}{12Ln^2}, \frac{1}{L}\sqrt{\frac{7}{27n+12}}\right\} \tag{3.10}$$

Then, in terms of the number of gradient computations c, the rate of convergence in expectation is

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_{t(c)})\right] \leq \max\left\{\frac{8\Omega k}{\gamma_{\max} c}, 16\sqrt{\frac{2\Omega n k\sigma^2}{c}}\right\}. \tag{3.11}$$

Remark that in Theorems 3.8, 3.9 and 3.10 the stepsizes chosen are constant and they are dependent on the total number of iterations t. We can get arbitrary precision selecting t and tuning γ, but there is no convergence at constant stepsizes in the non-smooth or smooth noisy cases. Looking at equations (3.8) and (3.11), there is convergence for stochastic extra-gradient under the smooth case with $\sigma^2 = 0$, both for stochastic extra-gradient and for doubly stochastic with variance reduction.

The results from Theorem 3.8 yield the first row of Table 3.1[3]. Bounds (3.9) and (3.11) yield the results in the last row. From Table 3.1 we compare the theoretical performance of doubly stochastic extra-gradient with stochastic extra-gradient.

i Under (**A**), Algorithm 2 performs with a similar rate to stochastic extra-gradients and the effect of the noise bound σ^2 is smaller.

ii Under (**B**), Algorithm 2 makes the $O(1/c)$ term of the bound $\sqrt{n/k}$ larger but it makes the $O(1/\sqrt{c})$ noise term $\sqrt{n/k}$ smaller. This is good in the long run, because $O(1/\sqrt{c})$ dominates.

Importance sampling. Using importance sampling allows to reduce the constant G in the non-stochastic part of the rate of convergence. Namely, we go from G to $\tilde{G}$ in Theorem 1 and 2. See Appendix for details.

[3]For simplicity, we overlook the dependency on constant factors and bound maxima by the sums of the two terms.

Chapter 4

Planning with Arithmetic and Geometric Attributes

4.1 Introduction

We consider an agent in a Markovian environment that is partially specified by a set of human-defined attributes. That is, each state of the environment is mapped to one attribute through a non-injective function. The attributes provide a useful language for communicating tasks to the agent. For example, in a task where an agent has to navigate in a maze and collect different items, useful attributes are the number of items collected, thus abstracting away objects locations.

In [53] it was shown that using a (state, attribute) framework it is possible for an agent to explore the environment and accomplish tasks at test time described by those attributes that were never seen during training. However, in that work, most of the attributes were computed through binary functions of the state, and the worst case complexity scaled exponentially in the number of attributes. Avoiding such curse of dimensionality therefore requires exploiting some underlying regularity in the space of attributes.

Many environments of interest have geometric and/or arithmetic structure. The simplest structure comes from the translation invariance of many environments with respect to its attributes: in a navigation task where attributes correspond to floors in a building, the task of going up or down is mostly independent of the current floor. We formulate our *Simple Structured Attribute Model* based on this inductive bias by considering a finite set of attribute differences or transitions. In many environments, the task of transitioning from one attribute to another is not only similar with respect to the attribute state, but may be regular *across* different transitions in attribute space. In other words, the task of moving in direction θ or θ' are similar if $\theta \approx \theta'$.

We leverage this further source of regularity inherited in the space of attribute differences in what we call our *Structured Attribute Model*.

We show that equipping attributes with the appropriate geometric and arithmetic structure brings substantial gains in sample complexity. Both Structured Attribute models can be deployed under mild assumptions, and provide different tradeoffs of sample complexity and prior structure. Whereas the Simple Structured Model is more robust to regularity assumptions on the tasks within the attribute space, it is less sample efficient than the Structured Attribute Model. On the other hand, our general Structured Attribute Model makes predictions of viable transitions in attribute space that may have never been seen during training. As our experiments demonstrate, in environments where attribute transitions lack enough regularity, these predictions may introduce unstabilities that ultimately impact performance.

We demonstrate the efficiency of both Structured Attribute Models on several 2d grid-world environments and unit-building tasks in StarCraft.

4.2 Problem Setup

We start with a Markovian Enviroment (ME) $(\mathcal{S}, \mathcal{A}, P)$, given by a state space $\mathcal{S}$, an action space $\mathcal{A}$ and a transition kernel $\mathcal{S} \times \mathcal{A} \to \mathcal{S}$ specified by the probability $P(s'|a, s)$ to transition from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ by taking action $a \in \mathcal{A}$. Model-based approaches attempt to estimate the transition kernel in order to perform planning. In this context, it is crucial to exploit regularity priors: in many practical scenarios this ME is highly structured, in the sense that the transition kernel varies smoothly with respect to specific transformations in the state/action spaces. For example, applying a force $a = \vec{F}$ to an object at location $\vec{l}$ will likely produce the same effect $\vec{l'}$ than applying the same action to the same object at a different location: $P(\vec{l'}|\vec{l}, \vec{F}) \approx P(\vec{l'} + \vec{l_0}|\vec{l} + \vec{l_0}, \vec{F})$

For that purpose, the ME is augmented with a *structured attribute space* $\mathcal{R}$ and a deterministic mapping $f : \mathcal{S} \to \mathcal{R}$, encoding the attributes $\rho = f(s)$ associated to each state. This mapping may be either given by the user, or may be regressed from a dataset of labeled pairs $\{(s_i, \rho_i)\}_i$, resulting in an estimate $\hat{f}$. Unless otherwise specified, in the following we shall write f to denote the ground-truth state-attribute mapping. In order to leverage the regularity of the environment, we equip the attribute space $\mathbb{R}$ with predefined algebraic and geometric structure. In this work, we consider attribute spaces built as outer products of elementary groups and monoids[1], such as real numbers $\mathbb{R}$, integers $\mathbb{Z}$, counts $\mathbb{N}$ and modular arithmetic $\mathbb{Z}/(q\mathbb{Z})$. If $\rho_1, \rho_2 \in \mathbb{R}$ are attributes, we will refer to $\rho_1 - \rho_2$ as an *attribute difference*. Our model-based approach thus amounts to estimating the transition kernel induced in the attribute space. At

[1] A monoid is a semigroup with identity element; a semigroup is a set with an associative binary operation.

test time, the agent will be given attribute goals ρ_g, and its objective is to take an appropriate sequence of actions in the original environment to reach ρ_g.

We consider attribute spaces $\mathbb{R}$ built as direct products of groups or monoids. In this work, we consider natural arithmetic attributes $\mathbb{N}$, modular arithmetic $\mathbb{Z}/(q\mathbb{Z})$, and real-valued attributes $\mathbb{R}$. We note however that our methodology can be easily extended to more exotic algebraic and geometric structures, such as modular real-valued attributes S^1, rigid motions $SO(2)$ or dihedral groups. Essentially, we can consider any set in which the difference of two elements is well defined.

4.3 Related work

This work builds upon the unstructured attribute planning model from [53]. In that work, a Markovian state space was augmented with a set of binary attributes. The attributes were used as a means of organizing exploration and specifying target states. In that work, the agent was built from three components: (i) a neural-net based **attribute detector** $\hat{f}$, which maps states s to a set of attributes ρ, i.e. $\rho = \hat{f}(s)$. (ii) a neural net-based **policy** $\pi(s, \rho_g)$ which takes a pair of inputs: the current state s and attributes of an (intermediate) goal state ρ_g, and outputs a distribution over actions. (iii) a **transition table** $c_\pi(\rho_i, \rho_j)$ that records the empirical probability that $\pi(s_{\rho_i}, \rho_j)$ can succeed at transiting successfully from ρ_i to ρ_j in a small number of steps.

In this work, in addition to allowing binary attributes, we consider attributes with more algebraic structure. In addition, in our Structured Attribute Model we augment the transition table c_π with a parametric transition detector that takes into account the structure of the attributes. We refer the reader to the references in [53] for a more complete review of the literature this work is built upon; but will briefly highlight a a few especially relevant works. Because we add further structure to the attributes, this work moves the unstructured attribute planner closer to [54–57], which discuss MDPs that can be written in terms of objects and relations between those objects. However, this current work still focuses on the interface between the symbolic description of the underlying Markovian space and the actual space; and the symbolic description in terms of attributes with algebraic structure is an approximation.

4.4 The Simple Structured Attribute Model

In this section we describe our Simple Structured Attribute Model, depicted in Figure 4.1. The Simple Structured Attribute Model relies on the assumption that the possible attribute-changing actions of the game are known. Attribute-changing game actions refer to those that imply a

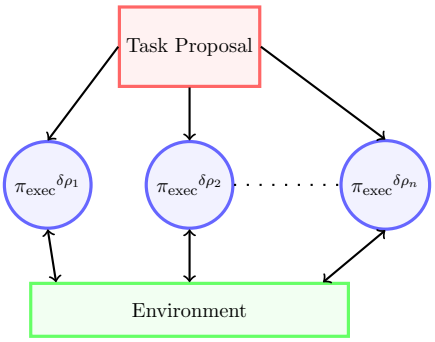

FIGURE 4.1: Block Diagram of the modules involved in the training of the Simple Structured Attribute Model.

change in the attribute space. For example, a "pick-up" action is attribute changing (if agent is correctly positioned), while "move left" is not attribute changing.

Given a game and an attribute space $\mathbb{R}$ for that game, if $\rho_1, \rho_2 \in \mathbb{R}$ are attributes, we denote $\rho_1 - \rho_2$ as an attribute difference. We define *one-step attribute differences* as attribute differences that correspond to a single attribute-changing action in the game. The distinction is particularly relevant for games in which the agent controls several units, such as Starcraft. In these games, several different units might take an attribute-changing action at the same time. We distinguish one-step attribute differences as those in which only one of these actions is taken.

The training of the model is divided in epochs, which divide themselves in batches and even further in episodes. Each episode is a succession of steps. At every step, the transition proposal module assigns an objective one-step attribute difference and the corresponding execution policy is used to determine the action to take. At the end of the batch, information about the steps and rewards is used to update the policies. Further details about each module are given below.

4.4.1 Execution Policies

All the attribute differences that are not one-step can be written as the sum of one-step attribute differences, as they correspond to the concatenation of several attribute-changing actions. Hence, we can restrict the planning path to contain only one-step attribute differences. For a given game and attribute space there is a fixed amount of one-step attribute differences, because the number of possible attribute-changing actions in the game is fixed and assumed to be known. That allows us to train a different execution policy for each one-step attribute difference.

For each one-step attribute difference $\delta\rho$, the execution policy $\pi_{\text{exec}}(\delta\rho)$ is parameterized by a neural network that takes as input the current state of the game and outputs a distribution over the game actions set $\mathcal{A}$. During training and at test time, this distribution is sampled to determine the action to be taken by the agent. Execution policies are trained with a policy gradient method (Generalized Advantage Estimation). During training, the transition module

produces target one-step attribute differences. The execution policy corresponding to the target difference $\delta\rho$ is used to direct the agent until $\delta\rho$ is completed (which yields a positive reward), a different attribute transition takes place or a certain number of steps is reached (which yield no reward). The rewards, states and actions are stored in the batch corresponding to $\delta\rho$ and used to compute the update for the the policy $\pi_{\text{exec}}(\delta\rho)$. The rewards, states and actions collected for a target one-step attribute difference are not used to train the policies corresponding to the other one-step attribute differences.

4.4.2 Inference

For each test episode, an attribute is given as the goal. The path of one-step attribute differences to be followed by the agent can be computed with a breadth-first search using a queue. If the agent deviates from the path, the path is recomputed. Every time we need to compute the path, we initialize a queue with the attribute corresponding to the current game state. During the search, when the attribute ρ is removed from the front of the queue does not meet the goal, we check whether $\rho \rightarrow \rho + \delta\rho$ is a possible transition for each one-step attribute difference $\delta\rho$. We add $\rho + \delta\rho$ to the queue for each possible transition. Since $\delta\rho$ is a one-step transition vector, determining whether $\rho \rightarrow \rho + \delta\rho$ is possible is the same as determining whether the game action corresponding to $\delta\rho$ can be taken when the game is in a state corresponding to ρ, which follows from the rules of the game. In other words, we need to know which attribute-changing actions are possible for a given attribute. For the games we have tried (including Starcraft) this information is very simple and explained in Appendix C.2.

4.4.3 Transition Proposals

The transition proposal method involves a breadth-first search, starting from the attribute corresponding to the current state of the game. In this case, the breadth-first search is stopped at a certain depth and a target attribute is randomly chosen among the attributes seen during the search. The list of one-step transition differences used to reach the chosen target attribute will be used as the proposals to train the execution policies. Whenever one one-step transition difference fails or the chosen target attribute is reached, a new list of proposals is computed.

4.5 The Structured Attribute Model

In this section we describe our Structured Attribute Model. The main differences with respect to the Simple Model are: (i) the Transition Detector, a Neural Network that predicts whether an unseen attribute transition will be feasible under the environment or not; and (ii) a different

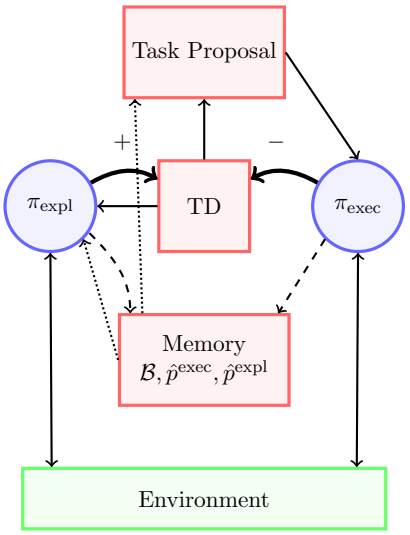

FIGURE 4.2: Block Diagram of the modules involved in the training of our Structured Attribute Model. Dashed lines correspond to Memory writes, dotted lines to memory reads, and thick lines provide labels to the Transition detector. Our model thus combines a non-parametric component given by the memory, and a parametric Transition Detector that extends the agent experience through the structured attribute space.

parametrization of the execution policy, which is fed as input the desired transition to be accomplished, therefore enforcing parameter sharing across different attribute transitions. The resulting model, depicted in Figure 4.2, contains several modules that interact with each other, with a structure similar to the unstructured model presented in [53]. The Structured Attribute Model does not rely on the assumption that the possible attribute-changing actions of the game are known: it works under the same assumption as the unstructured model, which is that we have an appropriate attribute space for our game. Hence, the Structured Model does not distinguish one-step attribute differences. The training of this model is analogous to the one for the simple model. However, in this case some steps are used to train an exploration policy, and at the end of each batch an update is done on the transition detector, the exploration policy and the execution policy. We describe each of the modules and their interactions below. As will be shown in the experiments, this model trades off exploration (and thus sample complexity) with an inductive bias on the similarities across tasks corresponding to different transitions.

4.5.1 Transition Detector

The transition detector is a neural network that receives as input an attribute ρ and an attribute difference $\delta\rho \in \mathbb{R} - \mathbb{R}$, and outputs $\text{TD}(\rho, \delta\rho)$, the estimated likelihood that with the current policy the transition will take place. This detector is trained in a supervised fashion by receiving both positive and negative samples. The positive samples are fed by the exploration policy π_{expl}, whereas the negative samples are produced by the execution policy π_{exec}.

4.5.2 Memory

The Structured Attribute Model contains two policies, detailed next: the exploration policy and the execution policy. In each case, we record the empirical counts on which attributes they have visited. Since the size of the attribute space grows exponentially with respect to the number of attributes, we consider only the marginal counts. For each attribute dimension $k \leq K$, we consider empirical marginal counts $\hat{p}_k^{\text{exec}}$, $\hat{p}_k^{\text{expl}}$ over each factor G_k of the attribute space $\mathcal{R}$. In case some attributes are continuous, we quantize them using a predefined number of bins in order to produce the empirical counts. Finally, in order to keep track of the admissible transitions in the full attribute space (without marginalization), we consider a memory buffer $\mathcal{B}$ that contains every observed attribute difference $\delta\rho$, this time without marginalization.

4.5.3 Exploration Policy

The estimation of the transition kernel starts with an exploration policy π_{expl} that looks for for transitions $\rho_i \to \rho_j$ in the attribute space. This policy is parameterized by a neural network that takes as input the pair $(s_i, \rho_i = f(s_i))$ and outputs a distribution over $\mathcal{A}$, which is then sampled to determine the action to be taken. The network is trained with policy gradient and its rewards are determined from the empirical marginal counts $\hat{p}^{\text{expl}}$. The reward for a transition $\rho_i \to \rho_j$ with $\delta\rho = \rho_j - \rho_i$ is $R_{\text{exp}} = -\alpha_1 \log \text{TD}(\rho_i, \delta\rho) + \alpha_2 g(\rho_i, \rho_j)$, $\text{TD}(\rho_i, \delta\rho)$ is the estimated probability that the transition will take place, which means that the first term rewards transitions that had been assigned low probability. The function $g(\rho_i, \rho_j)$ is inversely proportional to the times the exploration has previously seen this transition, estimated with the marginals of each component: $g(\rho_i, \rho_j) := \min_{k \leq K} \mathbf{1}(\rho_j[k] \neq \rho_i[k])(p_k^{\text{expl}}(\rho_j[k]))^{-0.5}$. The weighting hyper-parameters α_1, α_2 are adjustable to each environment and are reported in the experimental section.

4.5.4 Execution Policy

The execution policy π_{exec} takes as input the current state-attribute pair $(s, \rho = f(s))$ as well as a target attribute difference $\delta\rho$ (which may or may not have been seen before) provided by the Transition Proposal module, and outputs a distribution over the actions $\mathcal{A}$. It is trained with Generalized Advantage Estimation in the same way as the execution policies for the Simple Structured Attribute Model. The difference is that in this case there is one network for all target attribute difference, which is why the target attribute difference must be included in the input.

4.5.5 Transition Proposals

When proposing transitions, we want to accomplish two objectives: (i) enforce that the coverage of the execution policy π_{exec} matches that of the exploration π_{expl}, and (ii) provide the Transition Detector with negative samples, i.e. transitions that are not admissible in the environment. We propose to sample the target transitions $\delta\rho$ from the current buffer $\mathcal{B}$ of recorded transitions as follows. First, we filter out the transitions $\delta\rho \in \mathcal{B}$ that are considered unlikely to exist according to the current Transition detector using a threshold $\text{TD}(\rho, b) < \epsilon$ (we pick $\epsilon = 0.1$ in all experiments); call $\mathcal{B}^+$ the remaining transitions. Then we consider a mixture that samples uniformly at random within $\mathcal{B}^+$ with probability s_0, and according to the following distribution with probability $1-s_0$: $\forall\, \delta\rho \in \mathcal{B}^+$, $p(\delta\rho|\rho) = \frac{\gamma(\delta\rho)}{\sum_{b\in\mathcal{B}^+}\gamma(\delta\rho)}$, where $\gamma(\delta\rho) = \frac{\min_k \hat{p}_k^{\text{expl}}((\rho+\delta\rho)[k])}{1+\min_k \hat{p}_k^{\text{exec}}((\rho+\delta\rho)[k])}$. In words, we look at the differences in marginal counts between $\hat{p}_k^{\text{expl}}$ and $\hat{p}_k^{\text{exec}}$ across the recorded transitions in the buffer, and sample more often those where exploration outpaces execution.

4.5.6 Inference

Our inference strategy at test time is analogous to the unstructured attribute work [53], except that our estimated probabilities to realize each transition are given by the Transition Detector. Specifically, we look for the path $(\rho_0, \rho_1, \ldots, \rho_K)$ from the start $\rho_0 = f(s)$ to our goal $g = \rho_K$ that minimizes the distance defined by the Transition Detector as: $d(\rho_0, \rho_K) = -\sum_{i=0}^{K-1} \log(\text{TD}(\rho_i, \rho_{i+1} - \rho_i))$ To do this we use Dijkstra's algorithm on the graph that starts at the point where the agent is and extends to other points in the attribute space by applying the transitions in the buffer, giving each Transition (p, q) the cost $-\log(\text{TD}(p, q-p))$.

4.6 Experiments

We report experiments on Grid-World games using the Mazebase environments [58] and on Starcraft.

The Grid-World games consist of 2D maps that vary dynamically at each episode, of size between 7 and 10 in each dimension, with a single agent that interacts with the environment. In Grid-World games, we train our structured model without access to the test-time tasks during a prespecified number of episodes. After training, given a target task, we perform planning using Djikstra as explained in Section 4.5.6. For simplicity, we assume the state-to-attribute mapping $\rho = f(s)$ is known in our reported experiments. We consider two baselines: (i) A reinforcement learning agent parameterized with the same neural network architecture as our execution policy, taking the state and goal attribute as inputs, trained using a curriculum that starts from nearby tasks and extends them to the evaluation goals, and (ii) the Unstructured Attribute Planner

	Modular Switches	Exchangeable Attr	Constrained Attr
RL+curriculum	13.6%	14%	0.2%
Unstructured Attribute Model	9.6%	88%	20.8%
Simple Structured Attribute Model	**94.9%**	**92.4%**	**40.1%**
Structured Attribute Model	**89.3%**	**93%**	**81.6%**

TABLE 4.1: Percentage of proposed tasks that have been successfully accomplished using a fixed budget of allowed steps on three different Grid-World Environments. We consider a budget of 150 steps for all models. In the RL setting, we train the model on the same conditions as faced during testing. In the attribute setting, all the training is agnostic to the tasks proposed at test-time. Each test performance shown is computed as the average of four independent runs.

from [53], in which we treat attributes as a set. When the environment contains continuous attributes, we round them to the nearest integer and use the resulting discrete space.

4.6.1 Modular Switches

The first environment consists in 2-d mazes containing a variety of different objects that are randomly placed Depending on the state of a switch, the agent is allowed to pick objects of a specific kind, as illustrated in Figure C.1 in Appendix C.2. For each object type, we consider two attributes: how many objects are still available in the map, and how many objects the agent already collected. The attribute space is thus modeled with $\mathbb{R} = \underbrace{\mathbb{N} \times \dots \mathbb{N}}_{2q \text{ times}} \times (\mathbb{Z}/q\mathbb{Z})$, where q corresponds to the number of different objects. The last term $\mathbb{Z}/q\mathbb{Z}$ corresponds to the state of the switch. We consider $q = 3$ in our experiment. This environment is highly structured: denoting by $(a; b)$ an attribute with a number of items a and switch state b, the only admissible transitions are of the form $\delta\rho = (e_j; 0)$ with $e_j[j] = 1, e_j[j+q] = -1$, $j \leq q$, and $\delta\rho = (0; 1)$. The evaluation tasks consist in requesting a specific number of items of each category $\rho_{\text{target}} = (n_1, \dots, n_q)$. The RL version is trained with a curriculum that grows the distance (induced by the transitions in the attribute space) between the start and the end of the task, from 1 to 15. In this environment, Table 4.1 shows that neither the curriculum RL agent nor the Unstructured Attribute Planner model are able to successfully complete the target goals. The number of transitions is large relative to the number of steps. RL trained for 50M steps. Our two models as well as the Unstructured Attribute Model are trained for 25M steps for exploration and 25M steps for execution (only execution training for the Simple Model). Figure C.2 in Appendix C.2 displays the positive (resp. negative) transitions discovered by π_{expl} (resp. π_{exec}) as training progresses for the Structured Model. Our models are able to quickly generalize the transition kernels of the environment to unseen regions, by leveraging its rich arithmetic structure.

4.6.2 Exchangeable Attributes

This environment contains objects of several types, and for each type we consider two attributes: how many objects are still in the map, and how many objects the agent has already collected. At any time, the agent has the possibility to trade objects using pre-specified exchange rates, as shown in Figure C.1 in Appendix C.2. The attributes of this environment can be modeled as $\mathbb{R} = \underbrace{\mathbb{N} \times \ldots \mathbb{N}}_{2q \text{ times}}$, but this time the admissible transitions create interactions between attributes. The exchange rates $e_{i,j}$ and $f_{i,j}$ determine transitions of the form $(\rho[i], \rho[j]) \mapsto (\rho[i] - e_{i,j}, \rho[j] + f_{i,j})$, for $i, j = 1 \ldots q$. Inspired by real markets, we set $e_{i,j} > f_{i,j}$ for all pairs. The evaluation tasks consist in obtaining a predefined number of items of each type. We consider as before the case $q = 3$. The RL with curriculum is trained in 40M steps. Our model trained in 20M steps for exploration and 20M steps to train the policy. Similarly as before, the RL curriculum is implemented by growing the distance (induced by the transitions in the attribute space) between the start and the end of the task, from 1 to 30. In this case, Table 4.1 shows that, while our Structured and Simple Structured Models still outperform the two baselines, the difference is less dramatic than in other environments. We attribute this to the fact that the transition kernel is more homogeneous and faster mixing than before, and therefore although the exploration in the unstructured attributes misses many transitions, the planning phase manages to cover the attribute space more efficiently than in the modular switch environment.

4.6.3 Constrained Attributes

We consider an environment with continuous attribute components. Here, an agent is deployed in a terrain and it collects minerals with a single-use hammer. Each time the hammer is used, the agent receives a random amount of mineral coming from a continuous distribution over $\mathbb{R}_+$. The agent can go to the hardware store and obtain a new hammer in order to keep mining. He can also go to the dump yard and throw away a fixed amount of mineral. In that case, the attribute space is modeled as $\mathbb{R} = \underbrace{\mathbb{R} \times \ldots \mathbb{R}}_{q \text{ times}} \times \mathbb{N}$, where q is the number of different types minerals and each copy of $\mathbb{R}$ are is the amount of mineral of each type that the agent has. The admissible transitions are of the form $v = (-e_j; 0)$ (throwing mineral j to the dump yard) or $v = (Ce_j; -1)$ (mining at the mine for mineral j), where $C \sim \text{Unif}$, or $v = (\mathbf{0}; +1)$ (getting a new hammer). In order to make planning more challenging, we require that $\rho_{i+1} \in \mathcal{R}'$, where $\mathcal{R}'$ is a subset of the attribute space $\mathcal{R}$ that is fixed over all runs. Figure C.3 in Appendix C.2 illustrates the attribute space $\mathcal{R}$ and the subset $\mathcal{R}'$.

In this case when building the RL curriculum, the first stage are goals with high probability of being at just one transition away from the start, and next stages are defined by the Euclidean distance in the attribute space between the start and the end of the task. As with other

environments, at test-time the goals are to reach a certain point in attribute space. Since being close to the boundary reduces the probability of executing a transition, good plans in this environment need to traverse the attribute space while consistently staying inside of $\mathcal{S}$.

Table 4.1 shows that both our Structured Attribute Model is able to leverage the geometric structure of the environment to significantly outperform both RL and unstructured attribute baselines. The Simple Structured Attribute Model performs better than the two baselines, but much worse than the Structured Model. The reason is that the path planning method in the Simple Model is does not learn to stay inside of $\mathcal{S}$. We have tried using the Simple Structured Model with the path inference from the Structured Model (that is, training a transition network and using Dijkstra for planning but keeping a different execution network for each one-step attribute difference). In that case we obtain a test performance of 86.5%, which is higher than with the Structured Model. This increase is due to the better performance of the execution policies.

4.6.4 Starcraft

For the Starcraft experiments, we have used the same setup as in [53], which we will consider the baseline. We have used "StarCraft: Brood War" [59] and we consider the space of tasks of building particular units in a fixed time of 500 steps. We have restricted the game to the Terran race and only allow construction of certain units. Specifically, the agent can mine ore and build SCVs, supply depots, barracks, and marines. The attributes used are the amount of ore and the number of units of each type. For the case of Barracks, we have one component which indicates the number of Barracks currently under construction and another component indicating how many are finished. Thus, attributes for Starcraft are $(N_{\text{ore}}, N_{\text{SCV}}, N_{\text{built Barracks}}, N_{\text{marine}}, N_{\text{depot}}, N_{\text{unfinished Barracks}})$. The attribute space is modeled as $\mathbb{R} = \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}$. Unlike in [53], we use the full ore counts because we can deal with a bigger set of attributes.

The test tasks are the sames as in [53]. They are specified by attributes for which each component is randomly in the following way: the number of Marines is chosen between 0 and 4 with equal probability, the number of Barracks is chosen between 0 and 3 (unless the number of Marines is larger than 1, in which we choose between 1 and 3) and the number of Supply Depot is chosen between 0 and 3.

The Structured Attribute Model does not perform well for Starcraft. The average test performance remains below 5% indefinitely.

The Simple Structured Attribute Model does perform well for Starcraft, as shown in 4.3. After 7.5M steps, the average test success rate for 6 independent runs is 39.0%, although it has been

consistently over 40% and even over 50% at some epochs. That is, the agent was able to complete 39.0% of all proposed test goals. In comparison, the average test performance reported in [53] with the same test goal selection was 31.7%.

Two possible reasons why the Structured Attribute Model does not work well for Starcraft:

- The execution policy is harder to train for Starcraft than for the Mazebase games, as Starcraft is a more difficult game because that the agent must control a variable number of units (up to 15 in our setup).
- The inference strategy at test time in the Structured Attribute Model is unstable for Starcraft. The output of the transition network is a noisy estimate of the actual transition probability, and it is more sample-efficient than the attribute detector from [53] but less robust. This causes the planned paths to be unnecessarily long and usually increasing in length as training goes on. The specific reason is that mining ore is a transition completed with very high probability, as it is the most frequently seen. Hence, Dijkstra's algorithm will add some unnecessary transitions of this type in order to place the "hard" transitions between pairs of attributes that have been arbitrarily assigned particularly high probabilities by the transition network, even if the "hard" transition is equally likely between other pairs of attributes.

The Simple Structured Attribute Model addresses both issues, thus overperforming [53] while using less samples. We trained our model for 7.5M steps, and it reached the level of success rate of the unstructured model in 4M steps. The number of steps used in [53] was 30M (16M for exploration and 14M for execution). Since we are using an independent execution network for each one-step attribute difference, backpropagating samples corresponding to a given one-step attribute difference does not affect or worsen the output of the execution policy for all other one-step attribute differences. In the Structured Model, backpropagation of samples for a one-step attribute difference does have an impact on the output of the network for all the other simple attribute differences, because the parameters are shared. We have thus removed one source of noise. We have also removed the instability coming from the path inference. The inference strategy in the Simple Structured Attribute Model yields paths without superfluous transitions.

4.7 Discovery of Simple Attribute Differences

The Simple Structured Attribute Model relies on the list of one-step attribute differences. We propose a method to infer one-step transition differences. We run a certain number of steps

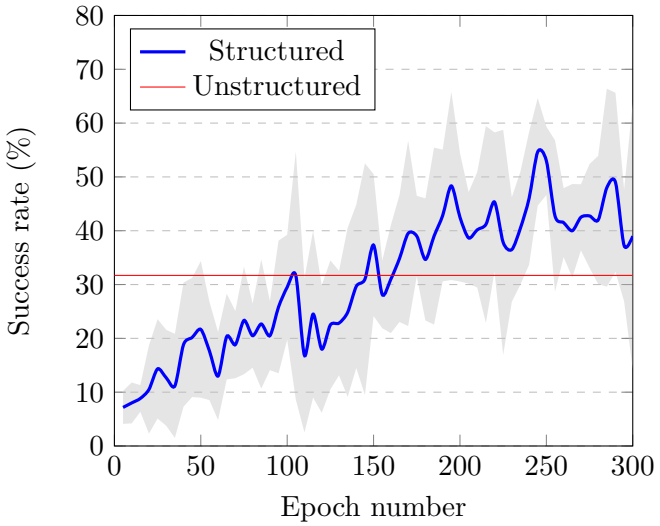

FIGURE 4.3: Success rate (%) of the Simple Structured Attribute Model for Starcraft. The blue line shows the average performance for 6 independent runs over 300 epochs. The light blue intervals show the confidence intervals corresponding to the standard deviations of the 6 runs. The red line shows the average test performance reported in [53]. One epoch contains 2500 steps for each thread (we use 10 threads), 300 epochs are 7.5M steps. For each of the runs, 100 test episodes were run every 5 epochs to compute test performance.

T using the exploration module from the Structured Attribute Model (while training the exploration policy). We store the attribute differences seen in a memory buffer $\mathcal{B}$, which we suppose contains all possible attribute differences. We want to find a minimal subset $\mathcal{C} \subseteq \mathcal{B}$ such that $\forall x \in \mathcal{B}, x = \sum_{i=1}^{N} n_i c_i$, where n_i are non-negative and $c_i \in \mathcal{C}$. In general this minimal subset is not uniquely defined. For example, for the set of linear combinations with non-negative integer coefficients of $(1,0),(2,3),(-2,-3)$, both the subsets $\{(1,0),(2,3),(-2,-3)\}$ and $\{(3,3),(2,3),(-2,-3)\}$ fulfill the conditions. This means that an exact identification of the set of one-step transition differences is not possible.

However, we observe one-step transition differences are expected to be seen more often than non-one-step transition differences and we can leverage this fact. Let K be the cardinality of the buffer $\mathcal{B}$ of attribute differences and $v_1, ..., v_K$ the elements of $\mathcal{B}$ sorted decreasingly by the number of times they have been seen during the exploration: v_1 is the most seen difference. Let $\{w_1, ..., w_{K'}\}$ be the set of one-step attribute differences of the game. We make the first assumption (i) that none of the one-step attribute differences can be expressed as a linear combination with non-negative integer coefficients of the other attribute differences. For each i between 1 and K, we can write $v_i = \sum_{j=1}^{K'} n_{i,j} w_j$, although the $n_{i,j}$ coefficients are not necessarily unique (because there might be integer linear combinations of the w_j that are equal to 0). Our second assumption is (ii) that $\{w_1, ..., w_{K'}\} \subseteq \{v_1, ..., v_K\}$ and for all j between 1 and K' we write $w_j = v_{f(j)}$. Coefficients $n_{i,f(j)}$ are unique and equal to δ_{ij} by assumption (i). We translate our observation into our third assumption: (iii) that $n_{i,j} = 0, \forall i < f(j)$, for any choice of the

$n_{i,j}$ coefficients. That is, for each one-step attribute difference w_j, the most seen difference that involves the attribute-changing action corresponding to w_j is w_j itself.

We use the following algorithm: (1) We initialize the set $\mathcal{C}$ as $\{v_1\}$, (2) For i from 2 through K, if v_i cannot be expressed as a linear combination of $v_1, ...v_{i-1}$ with non-negative integer coefficients, we add v_i to $\mathcal{C}$. The resulting set $\mathcal{C}$ is our guess of the set of one-step attribute differences, and it is a correct guess under the three assumptions stated in the previous paragraph.

Proof of correctness: $\mathcal{C}$ contains v_1, which must be equal to a one-step attribute difference $w_{f^{-1}(1)}$ because otherwise by assumption (iii) $n_{1,j} = 0$ for all j, which implies $v_1 = 0$. If $v_i = w_{f^{-1}(i)}$ is a one-step attribute difference, by assumption (i) it cannot be expressed as a linear combination of $\{w_1, ..., w_{K'}\} \setminus \{w_{f^{-1}(i)}\}$ with non-negative integer coefficients. By assumption (iii), each of the vectors $v_1, ..., v_{i-1}$ can be expressed as a linear combination of $\{w_1, ..., w_{K'}\} \setminus \{w_{f^{-1}(i)}\}$ with non-negative integer coefficients. Hence, v_i cannot be expressed as a linear combination of $v_1, ..., v_{i-1}$ with non-negative integer coefficients, which means that it will be added to $\mathcal{C}$. If $v_i = \sum_{j=1}^{K} n_{i,j} w_j$ is not a one-step attribute, then by (iii) for all j such that $n_{i,j} > 0$, $f(j) < i$, which means that $w_j \in v_1, ..., v_{i-1}$. This implies that v_i can be written as a linear combination of $v_1, ..., v_{i-1}$ with non-negative integer coefficients, which means that it will not be added to $\mathcal{C}$.

Over 5 independent runs of this discovery procedure in the Starcraft setup described in Subsection 4.6.4, it has been able to successfully identify the one-step attribute differences in all of them in less than 10 epochs (250K steps). It is robust, as the output set $\mathcal{C}$ was also correct after 50 epochs for all the runs.

A desirable property of an intelligent agent is its ability to understand its environment to quickly generalize to novel tasks and compose simpler tasks into more complex ones. If the environment has geometric or arithmetic structure, the agent should exploit these for faster generalization. Building on recent work that augments the environment with user-specified attributes, we show that further equipping these attributes with the appropriate geometric and arithmetic structure brings substantial gains in sample complexity.

Appendix A

Supplementary Material: Local convergence

A.1 Proof of Theorem 2.11

Lemma 2.10. If $A \in \mathbb{R}^{n\times n}$ is such that $\rho(A) < 1$, then there exists $C > 0$ such that for any x_0 the sequence $(x_k)_{k\in\mathbb{N}}$ in $\mathbb{R}^n$ defined as $x_{k+1} = Ax_k$ fulfills $||x_k||_2 \leq C||x_0||_2,\ \forall k \geq 0$.

Proof. Let $||\cdot||_{op}$ be the operator norm corresponding to the 2-norm. By Theorem 2.9 we know that $\lim_{k\to\infty} A^k = 0$. That means that there exists $K > 0$ such that $||A^K||_{op} < 1$. Every $k \in \mathbb{N}$ can be written as $k = mK + n$ with n between 0 and $K - 1$. Hence, we can write $||x_k||_2 = ||A^k x_0||_2 \leq ||A^k||_{op}||x_0||_2 \leq ||A^K||^m_{op}||A||^n_{op}||x_0||_2 \leq ||A||^n_{op}||x_0||_2$ Hence, we can choose $C = \max_{n\in\{0:K-1\}} ||A||^n_{op}$. □

Theorem 2.11. If $A \in \mathbb{R}^{n\times n}$ and $\rho(A) < 1$, the sequence $(\tilde{x}_k)_{k\in\mathbb{N}}$ taking values in $\mathbb{R}^n$ and defined as

$$\tilde{x}_{k+1} = A\tilde{x}_k + f(\tilde{x}_k)$$

where $\lim_{x\to 0} \frac{f(x)}{|x|} = 0$, there is a neighborhood U of 0 such that $(\tilde{x}_k)_{k\in\mathbb{N}}$ converges to 0 when $\tilde{x}_0 \in U$.

Proof. Similarly to the proof of Lemma 2.10, the fact that $\lim_{k\to 0} A^k = 0$ by Theorem 2.9 implies that $\exists K > 0$ such that $||A^K||_{op} < 1/2$. Once we have chosen K, there exists $C = \max_{n\in\{0:K-1\}} ||A||^n_{op}$ such that for any x_0, the sequence $(x_k)_{k\in\mathbb{N}}$ in $\mathbb{R}^n$ defined as $x_{k+1} = Ax_k$ fulfills $||x_k||_2 \leq C||x_0||_2,\ \forall k \geq 0$. We observe that $C \geq 1$.

It is easy to see that $\tilde{x}_k = A^k\tilde{x}_0 + \sum_{i=0}^{k-1} A^{k-1-i} f(\tilde{x}_i)$.

Since $\lim_{x\to 0} \frac{f(x)}{||x||_2} = 0$, we know

$$\forall \epsilon > 0, \exists \delta > 0 \text{ st } ||x||_2 < \delta \implies \frac{||f(x)||_2}{||x||_2} < \epsilon$$

Let $B_\delta(0)$ be the ball of radius δ centered at 0. Let $K' \in \mathbb{N}$ such that for all k with $0 \leq k \leq K'$ and for all $\tilde{x}_0 \in B_{\frac{\delta}{2C}}(0)$, we have $\tilde{x}_k \in B_\delta(0)$. K' exists because $K' = 0$ is valid. For all k smaller or equal than K' and any $\tilde{x_0} \in B_{\frac{\delta}{2C}}(0)$, we have

$$||f(\tilde{x}_k)||_2 < \epsilon ||\tilde{x}_k||_2 < \epsilon\delta$$

Hence, for all k smaller or equal than $K'+1$ and all $\tilde{x_0} \in B_{\frac{\delta}{2C}}(0)$,

$$\begin{aligned} ||\tilde{x}_k||_2 = ||A^k\tilde{x}_0 + \sum_{i=0}^{k-1} A^{k-1-i} f(\tilde{x}_i)||_2 &\leq ||A^k\tilde{x}_0||_2 + \sum_{i=0}^{k-1} ||A^{k-1-i} f(\tilde{x}_i)||_2 \\ &\leq ||A^k\tilde{x}_0||_2 + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op} ||f(\tilde{x}_i)||_2 \leq ||A^k\tilde{x}_0||_2 + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}\epsilon\delta \end{aligned} \quad \text{(A.1)}$$

Now, by Lemma 2.10, $||A^k\tilde{x}_0||_2 \leq C||\tilde{x}_0||_2 < C\frac{\delta}{2C} = \frac{\delta}{2}$, where we have also used $\tilde{x}_0 \in B_{\frac{\delta}{2C}}(0)$. Hence,

$$||A^k\tilde{x}_0||_2 + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}\epsilon\delta < \frac{\delta}{2} + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}\epsilon\delta$$

We consider the following inequality

$$\frac{\delta}{2} + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}\epsilon\delta < \delta \iff \frac{1}{2} + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}\epsilon < 1$$

By choosing $\epsilon > 0$ small enough we can get it to hold, which allows us to state that for all k with $0 \leq k \leq K'+1$ and all $x_0 \in B_{\frac{\delta}{2C}}(0)$, we have $\tilde{x}_k \in B_\delta(0)$. Of course, a choice of ϵ specifies a choice of δ and hence we cannot think of δ as a fixed quantity. Iterating this argument for a finite number of times (specifically $K - K'$ times, if $K - K'$ is positive), we conclude that with ϵ small enough and with the corresponding δ, for all k with $0 \leq k \leq K$ and all $x_0 \in B_{\frac{\delta}{2C}}(0)$, we have $\tilde{x}_k \in B_\delta(0)$.

If we set $k = K$, we can write

$$\begin{aligned} ||\tilde{x}_K|| \leq ||A^k\tilde{x}_0||_2 + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}\epsilon\delta \leq ||A^K||_{op}||\tilde{x}_0||_2 + \sum_{i=0}^{K-1} ||A^{K-1-i}||_{op}\epsilon\delta \\ < ||A^K||_{op}\frac{\delta}{2C} + \sum_{i=0}^{K-1} ||A^{K-1-i}||_{op}\epsilon\delta \end{aligned} \quad \text{(A.2)}$$

Taking into account that $||A^K||_{op} < 1/2$, we get $||A^K||_{op} < \frac{\delta}{4C}$. We can once again choose $\epsilon > 0$ small enough that

$$||A^K||_{op}\frac{\delta}{2C} + \sum_{i=0}^{K-1} ||A^{K-1-i}||_{op}\epsilon\delta < \frac{\delta}{4C} \iff ||A^K||_{op}\frac{1}{2C} + \sum_{i=0}^{K-1} ||A^{K-1-i}||_{op}\epsilon < \frac{1}{4C} \quad \text{(A.3)}$$

When the inequality holds, we have $||\tilde{x}_K|| < \frac{\delta}{4C}$. We can now prove by induction that keeping the same values of ϵ and δ, $||\tilde{x}_{NK}|| < \frac{\delta}{2^{N+1}C}$ and $||\tilde{x}_{NK+i}|| < \frac{\delta}{2^N}$ for i between 0 and $K-1$. This will conclude the proof. We need to prove the induction step. Suppose $||\tilde{x}_{NK}|| < \frac{\delta}{2^{N+1}C}$.

$$||\tilde{x}_{N(K+1)}|| = ||A^k\tilde{x}_{NK} + \sum_{i=0}^{k-1} A^{k-1-i}f(\tilde{x}_{NK+i})||_2 \leq ||A^k\tilde{x}_{NK}||_2 + \sum_{i=0}^{k-1} ||A^{k-1-i}f(\tilde{x}_{NK+i})||_2 \leq$$
$$||A^k\tilde{x}_{NK}||_2 + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}||f(\tilde{x}_{NK+i})||_2 = ||A^k\tilde{x}_{NK}||_2 + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}\epsilon\frac{\delta}{2^N} \leq$$
$$||A^K||_{op}||\tilde{x}_{NK}||_2 + \sum_{i=0}^{K-1} ||A^{K-1-i}||_{op}\epsilon\frac{\delta}{2^N} < ||A^K||_{op}\frac{\delta}{2^{N+1}C} + \sum_{i=0}^{K-1} ||A^{K-1-i}||_{op}\epsilon\frac{\delta}{2^N} \quad \text{(A.4)}$$

We have used $||f(\tilde{x}_{NK+i})||_2 < \epsilon||\tilde{x}_{NK+i}|| < \epsilon\frac{\delta}{2^N}$ for all i between 0 and $K-1$. We know that $||\tilde{x}_{NK+i}|| < \frac{\delta}{2^N}$ for all i between 0 and $K-1$ because we can reproduce the exact same argument with K' (using variable i instead of k) and we end up with

$$\frac{\delta}{2^{N+1}} + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}\epsilon\frac{\delta}{2^N} < \frac{\delta}{2^N} \iff \frac{1}{2} + \sum_{i=0}^{k-1} ||A^{k-1-i}||_{op}\epsilon < 1$$

for all k up to K, which are exactly the same conditions that ϵ has been chosen to satisfy.

Going back to A.4, in order to finish the proof we need to show that $||\tilde{x}_{N(K+1)}|| < \frac{\delta}{2^{N+2}C}$, for which suffices

$$||A^K||_{op}\frac{\delta}{2^{N+1}C} + \sum_{i=0}^{K-1} ||A^{K-1-i}||_{op}\epsilon\frac{\delta}{2^N} < \frac{\delta}{2^{N+2}C}$$

And this inequality is exactly the same one as Equation (A.3), which we have also chosen ϵ to satisfy. □

Appendix B

Supplementary Material: Stochastic Opponent Extrapolation

B.1 Useful lemmas

The following two lemmas are proven in [11] and will be used throughout the proofs several times. They are stated in the same notation as Subsection **??**.

Lemma B.1. *Let z be a point in $\mathcal{X}$, let χ, η be two points in the dual $E^\star$, let $w = P_z(\chi)$ and $r_+ = P_z(\eta)$. Then*

$$\|w - r_+\| \leq \|\chi - \eta\|_* \ , \tag{B.1}$$

and for all u one has

$$D(u, r_+) - D(u, z) \leq \langle \eta, u - w \rangle + \frac{1}{2}\|\chi - \eta\|_*^2 - \frac{1}{2}\|w - z\|^2 \ . \tag{B.2}$$

Lemma B.2. *Let $\xi_1, \xi_2, \ldots$ be a sequence of elements of $E^\star$. Define the sequence $\{y_\tau\}_{\tau=0}^\infty$ in $\mathcal{X}$ as follows:*

$$y_\tau = P_{y_{\tau-1}}(\xi_\tau), \quad y_0 \in Z^o.$$

Then y_τ is a measurable function of y_0 and $\xi_1, \ldots, \xi_\tau$ such that:

$$\forall u \in Z, \qquad \left\langle \sum_{\tau=1}^t \xi_t, y_{\tau-1} - u \right\rangle \leq D(u, y_0) + \frac{1}{2}\sum_{\tau=1}^t \|\xi_\tau\|_*^2. \tag{B.3}$$

The following two results are well known, but we will provide their proofs for completeness.

Lemma B.3. *Let $F : E \mapsto E^\star$ be a monotone operator. Let $(z_\tau)_{\tau\in[t]} \in \mathcal{X}$ and $\gamma_\tau \in (0,\infty)$ for τ between 0 and t. Let*

$$\hat{z}_\tau = \frac{\sum_{\tau=0}^t \gamma_\tau z_\tau}{\sum_{\tau=0}^t \gamma_\tau} \tag{B.4}$$

Then,

$$\mathrm{Err}_{VI}(\hat{z}_t) := \sup_{u\in Z} \langle F(u), \hat{z}_t - u\rangle \leq \sup_{u\in Z} \left(\sum_{\tau=0}^t \gamma_\tau\right)^{-1} \sum_{\tau=0}^t \langle \gamma_\tau F(z_\tau), z_\tau - u\rangle. \tag{B.5}$$

Proof. By monotonicity of F,

$$\langle F(u), \hat{z}_t - u\rangle = \left(\sum_{\tau=0}^t \gamma_\tau\right) \sum_{\tau=0}^t \langle \gamma_\tau F(u), z_\tau - u\rangle \leq \left(\sum_{\tau=0}^t \gamma_\tau\right) \sum_{\tau=0}^t \langle \gamma_\tau F(z_\tau), z_\tau - u\rangle. \tag{B.6}$$

□

Lemma B.4. *Let the game defined by losses ℓ_i, $1 \leq i \leq n$ be a convex n-player game. Let $(z_\tau)_{\tau\in[t]} \in \times$ and $\gamma_\tau \in (0,\infty)$ for τ between 0 and t. Let $F : \mathbb{R}^{nd} \to \mathbb{R}^{nd}$ be the simultaneous (sub)differential. Let $\hat{z}_\tau$ as in Lemma B.3. Then,*

$$\mathrm{Err}_N(\hat{z}_t) := \sup_{u\in Z} \sum_{i=1}^n \ell_i(\hat{z}_t) - \ell_i(u^{(i)}, \hat{z}_t^{(-i)}) \leq \sup_{u\in Z} \left(\sum_{\tau=0}^t \gamma_\tau\right)^{-1} \sum_{\tau=0}^t \langle \gamma_\tau F(z_\tau), z_\tau - u\rangle. \tag{B.7}$$

Proof. By the definition of convex n-player game,

$$\begin{aligned}\sum_{i=1}^n \ell_i(\hat{z}_t) - \ell_i(u^{(i)}, \hat{z}_t^{(-i)}) &\leq \left(\sum_{\tau=0}^t \gamma_\tau\right)^{-1} \sum_{\tau=0}^t \gamma_\tau \sum_{i=1}^n \ell_i(z_\tau) - \ell_i(u^{(i)}, z_\tau^{(-i)}) \\ &\leq \left(\sum_{\tau=0}^t \gamma_\tau\right)^{-1} \sum_{\tau=0}^t \sum_{i=1}^n \langle \gamma_\tau \nabla_i \ell_i(z_\tau), z_\tau^{(i)} - u^{(i)}\rangle = \left(\sum_{\tau=0}^t \gamma_\tau\right)^{-1} \sum_{\tau=0}^t \langle \gamma_\tau F(z_\tau), z_\tau - u\rangle. \end{aligned} \tag{B.8}$$

□

Lemma B.5. *Let $(\gamma_t)_{t\in\mathbb{N}}$ be a sequence in $(0,\infty)$ and $A, B > 0$. For any $t \in \mathbb{N}$,*

$$f(\alpha) := \frac{A}{\sum_{\tau=0}^t \alpha\gamma_t} + \frac{B\sum_{\tau=0}^t (\alpha\gamma_t)^2}{\sum_{\tau=0}^t \alpha\gamma_t} \tag{B.9}$$

attains its minimum for $\alpha > 0$ when both terms are equal. The value at the minimum is

$$f(\alpha^\star) = f\left(\sqrt{\frac{A}{B\sum_{\tau=0}^t \gamma_t^2}}\right) = \frac{2\sqrt{AB\sum_{\tau=0}^t \gamma_t^2}}{\sum_{\tau=0}^t \gamma_t} \tag{B.10}$$

Proof. From the first order condition:

$$-\frac{1}{\alpha^2}\frac{A}{\sum_{\tau=0}^{t}\gamma_t}+\frac{B\sum_{\tau=0}^{t}\gamma_t^2}{\sum_{\tau=0}^{t}\gamma_t}=0 \tag{B.11}$$

and the result follows. □

Lemma B.6. *Let $X_1, X_2, ..., X_n$ be Banach spaces. Let $X = X_1 \times X_2 \times ... \times X_n$. Let us define a norm on X. If $y_1, ..., y_n \in X_1, ...X_n$,*

$$\|(y_1, ..., y_n)\|_X := \sqrt{\sum_{i=1}^{n}\|y_i\|_{X_i}^2} \tag{B.12}$$

It is known that $(X, \|\cdot\|_X)$ is a Banach space. Let $X^\star = X_1^\star \times X_2^\star \times ... \times X_n^\star$ be the dual of X and denote the dual norms by $\|\cdot\|_{X^\star}$ and $\|\cdot\|_{X_i^\star}$. Then, for any $a = (a_1, ..., a_n) \in X^\star$,

$$\|a\|_{X^\star}^2 \leq \sum_{i=1}^{n}\|a_i\|_{X_i^\star}^2 \tag{B.13}$$

Proof.

$$\begin{aligned}\|a\|_{X^\star}^2 = \sup_{y\in X}\frac{|ay|^2}{\|y\|_{X^\star}^2} = \sup_{y\in X}\frac{\left|\sum_{i=1}^{n}a_iy_i\right|^2}{\|y\|_{X^\star}^2} &\leq \sup_{y\in X}\frac{\left|\sum_{i=1}^{n}\|a_i\|_{X_i^\star}\|y_i\|_{X_i}\right|^2}{\|y\|_{X^\star}^2} \\ &\leq \sup_{y\in X}\frac{\left(\sum_{i=1}^{n}\|a_i\|_{X_i^\star}^2\right)\left(\sum_{i=1}^{n}\|y_i\|_{X_i}^2\right)}{\|y\|_{X^\star}^2} = \sum_{i=1}^{n}\|a_i\|_{X_i^\star}^2\end{aligned} \tag{B.14}$$

□

B.2 Mirror-prox with player stochasticity

B.2.1 Proof of convergence in the non-smooth case

Theorem B.7. *Let $(\hat{\theta}_t)_{t\in\mathbb{N}}$ given by Algorithm 3. Then,*

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_t)\right] \leq \left(\sum_{\tau=0}^{t}\gamma_\tau\right)^{-1}\left(2\Omega + \sum_{\tau=0}^{t}\gamma_\tau^2 n\left(\frac{3G^2}{p}+\sigma^2\right)\right) \tag{B.15}$$

Proof. The strategy of the proof is similar to the proof of Theorem 2 and part of Theorem 1 from [11]. We look for a bound of $\sum_{\tau=0}^{t}\langle\gamma_\tau F(\theta_{\tau+1/2}), \theta_{\tau+1/2} - u\rangle$, which by Lemma B.4 is itself a bound of the functional Nash error.

Algorithm 3 Mirror-prox with player stochasticity

Choose stepsizes $\gamma_\tau > 0$ for $1 \leq \tau \leq t$.
Choose initial value $x_0 \in \mathbb{R}^{nd}$.
for $\tau = 0, \dots, t$ **do**
    Sample the random matrices $M_\tau, M_{\tau+1/2} \in \mathbb{R}^{nd \times nd}$. Each diagonal block is the identity with probability p and 0 with probability $1-p$.
    Intermediate step: $\theta_{\tau+1/2} = P_{\theta_\tau}\left(\gamma_\tau \frac{M_\tau}{p} \hat{F}(\theta_\tau)\right) = P_{\theta_\tau}(\gamma_\tau \tilde{F}_{\tau+1/2})$.
    Extra-gradient step: $\theta_{\tau+1} = P_{\theta_\tau}\left(\gamma_\tau \frac{M_{\tau+1/2}}{p} \hat{F}(\theta_{\tau+1/2})\right) = P_{\theta_\tau}(\gamma_\tau \tilde{F}_{\tau+1}))$.
Return $\hat{\theta}_t$ where

$$\hat{x}_t = \left[\sum_{\tau=0}^{t} \gamma_\tau\right]^{-1} \sum_{\tau=0}^{t} \gamma_\tau \theta_{\tau+1/2}.$$

By using Lemma B.1 with $z = \theta_\tau$, $\chi = \gamma_\tau \tilde{F}_{\tau+1/2}$, $\eta = \gamma_\tau \tilde{F}_{\tau+1}$ (so that $w = \theta_{\tau+1/2}$ and $r_+ = \theta_{\tau+1}$), we have for any $u \in Z$

$$\begin{aligned}\langle \gamma_\tau \tilde{F}_{\tau+1}, \theta_{\tau+1/2} - u\rangle + D(u, \theta_{\tau+1}) - D(u, \theta_\tau) &\leq \frac{\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1} - \tilde{F}_{\tau+1/2}\|_\star^2 - \frac{1}{2}\|\theta_{\tau+1/2} - \theta_\tau\|^2 \\ &\leq \frac{\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1} - \tilde{F}_{\tau+1/2}\|_\star^2 \quad \text{(B.16)}\end{aligned}$$

When summing up from $\tau = 0$ to $\tau = t$ in equation (B.16), we obtain:

$$\sum_{\tau=0}^{t} \langle \gamma_\tau \tilde{F}_{\tau+1}, \theta_{\tau+1/2} - u\rangle \leq D(u, x_0) - D(u, x_{t+1}) + \sum_{\tau=0}^{t} \frac{\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1} - \tilde{F}_{\tau+1/2}\|_\star^2 \quad \text{(B.17)}$$

Remark that we can decompose the right-hand side and obtain:

$$\begin{aligned}\sum_{\tau=0}^{t} \langle \gamma_\tau F(\theta_{\tau+1/2}), \theta_{\tau+1/2} - u\rangle &\leq D(u, x_0) - D(u, x_{t+1}) + \sum_{\tau=0}^{t} \frac{\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1} - \tilde{F}_{\tau+1/2}\|_\star^2 \\ &\quad + \sum_{\tau=0}^{t} \left\langle \gamma_\tau (F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}), \theta_{\tau+1/2} - u \right\rangle = \Omega + \sum_{\tau=0}^{t} \frac{\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1} - \tilde{F}_{\tau+1/2}\|_\star^2 \\ &\quad + \gamma_\tau \sum_{\tau=0}^{t} \left\langle F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}, \theta_{\tau+1/2} - y_\tau \right\rangle + \gamma_\tau \sum_{\tau=0}^{t} \left\langle F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}, y_\tau - u \right\rangle, \quad \text{(B.18)}\end{aligned}$$

where $y_{\tau+1} = P_{y_\tau}(\gamma_\tau \Delta_\tau)$ with $y_0 = x_0$ and $\Delta_\tau = F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}$.

Equation (B.18) is also used in [11]. We are now going to bound the expectation of each of the right-hand side terms separately.

We define the filtration $\mathcal{F}'_\tau = \{\theta_0, \theta_{1/2}, \ldots, \theta_\tau\}$. For the second term on the right-hand side of (B.18), we take the expectation of $\|\tilde{F}_{\tau+1} - \tilde{F}_{\tau+1/2}\|_\star^2$:

$$\begin{aligned}
\mathbb{E}\left[\|\tilde{F}_{\tau+1} - \tilde{F}_{\tau+1/2}\|_\star^2\right] &\leq 2\left(\mathbb{E}\left[\|\tilde{F}_{\tau+1}\|_\star^2\right] + \mathbb{E}\left[\|\tilde{F}_{\tau+1/2}\|_\star^2\right]\right) \\
&= \frac{2}{p^2}\left(\mathbb{E}\left[\mathbb{E}\left[\|M_{\tau+1/2}\hat{F}(\theta_{\tau+1/2})\|_\star^2|\mathcal{F}_\tau\right]\right] + \mathbb{E}\left[\mathbb{E}\left[\|M_\tau\hat{F}(\theta_\tau)\|_\star^2|\mathcal{F}'_\tau\right]\right]\right) \\
&\overset{\text{Lemma B.6}}{\leq} \frac{2}{p^2}\sum_{i=1}^n\left(\mathbb{E}\left[\mathbb{E}\left[\|M^{(i)}_{\tau+1/2}\hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2|\mathcal{F}_\tau\right]\right] + \mathbb{E}\left[\mathbb{E}\left[\|M^{(i)}_\tau\hat{F}^{(i)}(\theta_\tau)\|_\star^2|\mathcal{F}'_\tau\right]\right]\right) \\
&\leq \frac{2}{p}\sum_{i=1}^n\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] + \mathbb{E}\left[\|\hat{F}^{(i)}(\theta_\tau)\|_\star^2\right] \leq \frac{4nG^2}{p} \qquad \text{(B.19)}
\end{aligned}$$

We define the filtration $\mathcal{F}_\tau = \{\theta_0, \theta_{1/2}, \ldots, \theta_\tau, \theta_{\tau+1/2}\}$. By taking the expectation with respect to this filtration, the third term of equation (B.18) becomes:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{\tau=0}^t\mathbb{E}\left[\left\langle\gamma_\tau\left(I - \frac{M_{\tau+1/2}}{p}\right)\hat{F}(\theta_{\tau+1/2}), \theta_{\tau+1/2} - y_\tau\right\rangle\middle|\mathcal{F}_\tau\right]\right] \\
&\quad= \mathbb{E}\left[\sum_{\tau=0}^t\left\langle\gamma_\tau\mathbb{E}\left[\left(I - \frac{M_{\tau+1/2}}{p}\right)\middle|\mathcal{F}_\tau\right]\mathbb{E}\left[\hat{F}(\theta_{\tau+1/2})\middle|\mathcal{F}_\tau\right], \theta_{\tau+1/2} - y_\tau\right\rangle\right] = 0 \qquad \text{(B.20)}
\end{aligned}$$

By using the sequences $\{y_\tau\}$ and $\{\xi_\tau = \gamma_\tau\Delta_\tau\}$ in Lemma B.2 as in [11], we obtain:

$$\sum_{\tau=0}^t\langle\gamma_\tau\Delta_\tau, y_\tau - u\rangle \leq D(u, x_0) + \sum_{\tau=0}^t\frac{\gamma_\tau^2}{2}\|\Delta_\tau\|_\star^2 \leq \Omega + \sum_{\tau=0}^t\frac{\gamma_\tau^2}{2}\|F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}\|_\star^2 \qquad \text{(B.21)}$$

We bound the expectation of $\|F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}\|_\star^2$ using the filtration $\mathcal{F}_\tau$:

$$\begin{aligned}
\mathbb{E}\left[\|F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}\|_\star^2\right] &\overset{\text{Lemma B.6}}{\leq} \sum_{i=1}^n \mathbb{E}\left[\|F^{(i)}(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}^{(i)}\|_\star^2\right] \\
&= \sum_{i=1}^n \mathbb{E}\left[\left\|F^{(i)}(\theta_{\tau+1/2}) - \frac{M_{\tau+1}^{(i)}}{p}\hat{F}^{(i)}(\theta_{\tau+1/2})\right\|_\star^2\right] \leq \sum_{i=1}^n 2\mathbb{E}\left[\left\|\left(I - \frac{M_{\tau+1}^{(i)}}{p}\right)\hat{F}^{(i)}(\theta_{\tau+1/2})\right\|_\star^2\right] \\
&\quad + \sum_{i=1}^n 2\mathbb{E}\left[\left\|F^{(i)}(\theta_{\tau+1/2}) - \hat{F}^{(i)}(\theta_{\tau+1/2})\right\|_\star^2\right] \\
&\leq \sum_{i=1}^n 2\mathbb{E}\left[p\left\|\frac{p-1}{p}\hat{F}^{(i)}(\theta_{\tau+1/2})\right\|_\star^2 + (1-p)\|\hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] + 2n\sigma^2 \\
&= \sum_{i=1}^n 2\left(1 - p + \frac{(1-p)^2}{p}\right)\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] + 2n\sigma^2 \\
&\leq \sum_{i=1}^n 2\frac{1-p}{p}\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] + 2n\sigma^2 = \frac{2nG^2(1-p)}{p} + 2n\sigma^2 = \frac{2nG^2}{p} + 2n(\sigma^2 - G^2)
\end{aligned} \tag{B.22}$$

Therefore, by taking the expectation in equation (B.18) and by using the upper bounds above, we finally get:

$$\mathbb{E}\left[\sup_{u\in Z}\sum_{\tau=0}^t \langle \gamma_\tau F(\theta_{\tau+1/2}), \theta_{\tau+1/2} - u\rangle\right] \leq 2\Omega + \sum_{\tau=0}^t \gamma_\tau^2 n\left(\frac{3G^2}{p} + \sigma^2 - G^2\right) \tag{B.23}$$

Lemma B.4 yields:

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_t)\right] \leq \left(\sum_{\tau=0}^t \gamma_\tau\right)^{-1}\left(2\Omega + \sum_{\tau=0}^t \gamma_\tau^2 n\left(\frac{3G^2}{p} + \sigma^2 - G^2\right)\right) \tag{B.24}$$

□

Corollary B.8. *If we set $\gamma_\tau = \alpha/\sqrt{\tau}$ for $t \geq \tau \geq 1$ (and $\gamma_0 = \alpha$) and we minimize the bound from Theorem B.7 with respect to α at a fixed time t, we get*

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_t)\right] \leq \frac{2\sqrt{2\Omega n\left(\frac{3G^2}{p} + \sigma^2\right)(2 + \log(t))}}{2\sqrt{t} - 1} \approx \sqrt{\frac{2\Omega n\left(\frac{3G^2}{p} + \sigma^2\right)\log(t)}{t}} \tag{B.25}$$

If we set $\gamma_\tau = \gamma$ for $t \geq \tau \geq 0$ and we minimize the bound from Theorem B.7 with respect to γ at a fixed time t, we get

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_t)\right] \leq \sqrt{\frac{8\Omega n\left(\frac{3G^2}{p} + \sigma^2 - G^2\right)}{t}} \tag{B.26}$$

Proof. For Equation (B.25), we use:

$$1+\sum_{\tau=1}^{t}\frac{1}{\sqrt{\tau}}\geq 1+\int_{\tau=1}^{t}\frac{1}{\sqrt{\tau}}=2\sqrt{t}-1 \tag{B.27}$$

$$2+\sum_{\tau=2}^{t}\frac{1}{\tau}\leq 2+\int_{\tau=1}^{t}\frac{1}{\tau}=2+\log(t) \tag{B.28}$$

We plug this on Theorem B.7 and we optimize over α using Lemma B.5. For Equation (B.26), we just use Lemma B.5. □

Remark B.9. The optimized bound with constant stepsizes decreases faster than the optimized bound with decreasing stepsizes. Constant stepsizes can achieve an arbitrarily low value of the bound at a certain t but do not ensure convergence. Stepsizes decreasing as $1/\sqrt{\tau}$ do ensure convergence.

B.2.2 Bounds in terms of the number of gradient computations (non-smooth case)

Let us restrict ourselves to the case in which the masks $M_\tau, M_{\tau+1/2}$ are chosen such that exactly k of the n diagonal blocks are the identity and all choices are equiprobable. In this case, $p=k/n$. The analysis can be carried out for other selections of $M_\tau, M_{\tau+1/2}$, but this one makes the argument particularly simple.

Let us define $c(t)$ as the number of gradients of the form $\nabla_i \ell_i$ computed up to iteration t. Since $2k$ gradients are computed at each iteration of the algorithm, $c(t)=2kt$. The inverse function is $t(c)=c/(2k)$. If we set $p=k/n$ and we put t in terms of c, the bounds in Equations (B.25) and (B.26) become:

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_{t(c)})\right]=\sqrt{\frac{2\Omega n\left(\frac{3G^2n}{k}+\sigma^2-G^2\right)\log(\frac{c}{2k})}{\frac{c}{2k}}}=\sqrt{\frac{4\Omega n\left(3G^2n+k(\sigma^2-G^2)\right)\log(\frac{c}{2k})}{c}} \tag{B.29}$$

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_{t(c)})\right]=\sqrt{\frac{8\Omega n\left(\frac{3G^2n}{k}+\sigma^2-G^2\right)}{\frac{c}{2k}}}=4\sqrt{\frac{\Omega n\left(3G^2n+k(\sigma^2-G^2)\right)}{c}} \tag{B.30}$$

B.2.3 Proof of convergence in the smooth case

The added assumption is that the gradients $\nabla_i \ell_i$ of the losses are L-Lipschitz, i.e. for all i between 1 and n,

$$\|\nabla_i \ell_i(x)-\nabla_i \ell_i(y)\|_\star \leq L\|x-y\|_2 \tag{B.31}$$

Alternatively, we can say that losses ℓ_i are L-smooth with respect to parameters $x^{(i)}$.

Theorem B.10. *Let $(\hat{\theta}_t)_{t\in\mathbb{N}}$ given by Algorithm 3. If each loss ℓ_i is L-smooth with respect to $x^{(i)}$ and for all τ, $\gamma_\tau < 1/(L\sqrt{3n})$, then*

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_t)\right] \leq \left(\sum_{\tau=0}^{t}\gamma_\tau\right)^{-1}\left(2\Omega + \sum_{\tau=0}^{t}\frac{13\gamma_\tau^2 n}{2}\left(\frac{G^2(1-p)}{p}+\sigma^2\right)\right) \tag{B.32}$$

Equation (B.33) from the proof of Theorem B.7 becomes:

$$\begin{aligned}
&\langle\gamma_\tau\tilde{F}_{\tau+1}, \theta_{\tau+1/2}-u\rangle + D(u,\theta_{\tau+1}) - D(u,\theta_\tau) \leq \frac{\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1}-\tilde{F}_{\tau+1/2}\|_\star^2 - \frac{1}{2}\|\theta_{\tau+1/2}-\theta_\tau\|_2^2 \\
&\leq \frac{3\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1}-F(\theta_{\tau+1/2})\|_\star^2 + \frac{3\gamma_\tau^2}{2}\|F(\theta_\tau)-\tilde{F}_{\tau+1/2}\|_\star^2 + \frac{3\gamma_\tau^2}{2}\|F(\theta_{\tau+1/2})-F(\theta_\tau)\|_\star^2 - \frac{1}{2}\|\theta_{\tau+1/2}-\theta_\tau\|_2^2 \\
&\leq \frac{3\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1}-F(\theta_{\tau+1/2})\|_\star^2 + \frac{3\gamma_\tau^2}{2}\|F(\theta_\tau)-\tilde{F}_{\tau+1/2}\|_\star^2 + \frac{3n\gamma_\tau^2 L^2}{2}\|\theta_{\tau+1/2}-\theta_\tau\|_\star^2 - \frac{1}{2}\|\theta_{\tau+1/2}-\theta_\tau\|_2^2 \\
&\leq \frac{3\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1}-F(\theta_{\tau+1/2})\|_\star^2 + \frac{3\gamma_\tau^2}{2}\|F(\theta_\tau)-\tilde{F}_{\tau+1/2}\|_\star^2
\end{aligned} \tag{B.33}$$

For the last inequality we have used that $\gamma_t \leq 1/(L\sqrt{3n})$. Now we use the reasoning from Equation (B.22) to get:

$$\mathbb{E}\left[\|\tilde{F}_{\tau+1}-F(\theta_{\tau+1/2})\|_\star^2\right] \leq \frac{2nG^2(1-p)}{p} + 2n\sigma^2 \tag{B.34}$$

$$\mathbb{E}\left[\|F(\theta_\tau)-\tilde{F}_{\tau+1/2}\|_\star^2\right] \leq \frac{2nG^2(1-p)}{p} + 2n\sigma^2 \tag{B.35}$$

Hence, using the equations above and Equation (B.21) we can write

$$\mathbb{E}\left[\sup_{u\in Z}\sum_{\tau=0}^{t}\langle\gamma_\tau F(\theta_{\tau+1/2}), \theta_{\tau+1/2}-u\rangle\right] \leq 2\Omega + \sum_{\tau=0}^{t}\frac{13\gamma_\tau^2 n}{2}\left(\frac{G^2(1-p)}{p}+\sigma^2\right) \tag{B.36}$$

And we use Lemma B.4 to conclude.

Corollary B.11. *For all τ such that $1 \leq \tau \leq t$, set*

$$\gamma_\tau = \gamma = \min\left\{\frac{1}{L\sqrt{3n}}, 2\sqrt{\frac{\Omega}{13n\left(\frac{G^2(1-p)}{p}+\sigma^2\right)t}}\right\} \tag{B.37}$$

Then,

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_t)\right] \leq \max\left\{\frac{4L\Omega\sqrt{3n}}{t}, 2\sqrt{\frac{13\Omega n\left(\frac{G^2(1-p)}{p}+\sigma^2\right)}{t}}\right\} \tag{B.38}$$

Proof. When the minimum is achieved by the second term the inequality holds by Lemma B.5. In the case $\gamma = 1/(L\sqrt{3n})$:

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_t)\right] \leq \frac{2L\Omega\sqrt{3n}}{t} + \left(\frac{1}{L\sqrt{3n}}\right)^2 \frac{13tn}{2}\left(\frac{G^2(1-p)}{p} + \sigma^2\right) \leq \frac{4L\Omega\sqrt{3n}}{t} \tag{B.39}$$

The second inequality holds because $1/(L\sqrt{3n}) < \alpha^\star$, which means that the first term is greater than the second one. □

B.2.4 Bound in terms of the number of gradient computations (smooth case)

If we write $p = k/n$, $t(c) = c/(2k)$ as in Subsection B.2.2 for Equation (B.38):

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_{t(c)})\right] \leq \max\left\{\frac{8kL\Omega\sqrt{3n}}{c}, 2\sqrt{\frac{26\Omega\left(G^2n(n-k) + \sigma^2nk\right)}{c}}\right\} \tag{B.40}$$

B.3 Mirror-prox with player stochasticity and importance sampling

Algorithm 4 Mirror-prox with player stochasticity and importance sampling

1: Choose stepsizes $\gamma_\tau > 0$ for $1 \leq \tau \leq t$.
2: Choose initial value $x_0 \in \mathbb{R}^{nd}$.
3: **for** $\tau = 0, \ldots, t$ **do**
4: Sample the random matrices $M_\tau, M_{\tau+1/2} \in \mathbb{R}^{nd\times nd}$. Each diagonal block i is the identity with probability $p_i = npG_i/\sum_{k=1}^n G_k$ and 0 with probability $1 - p_i$.
5: Define $\tilde{M}_\tau, \tilde{M}_{\tau+1/2} \in \mathbb{R}^{nd\times nd}$ dividing each diagonal block of $M_\tau, M_{\tau+1/2}$ by p_i.
6: Intermediate step: $x_{\tau+1/2} = P_{x_\tau}\left(\gamma_\tau \tilde{M}_\tau \hat{F}(x_\tau)\right) = P_{x_\tau}(\gamma_\tau \tilde{F}_{\tau+1/2})$.
7: Extra-gradient step: $x_{\tau+1} = P_{x_\tau}\left(\gamma_\tau \tilde{M}_{\tau+1/2}\hat{F}(x_{\tau+1/2})\right) = P_{x_\tau}(\gamma_\tau \tilde{F}_{\tau+1}))$.
8: Return $\hat{\theta}_t$ where

$$\hat{x}_t = \left[\sum_{\tau=0}^t \gamma_\tau\right]^{-1} \sum_{\tau=0}^t \gamma_\tau x_{\tau+1/2}.$$

Remark B.12. A sufficient and necessary condition for $p_i \leq 1$ for all i between 1 and n is

$$p \leq \frac{\sum_{k=1}^n G_k}{n \max_i G_i} \tag{B.41}$$

Theorem B.13. *Let $(\hat{\theta}_t)_{t\in\mathbb{N}}$ given by Algorithm 3. Then,*

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_t)\right] \leq \left(\sum_{\tau=0}^t \gamma_\tau\right)^{-1}\left(2\Omega + \sum_{\tau=0}^t \gamma_\tau^2\left(\frac{3(\sum_{i=0}^n G_i)^2}{np} + n\sigma^2\right)\right) \tag{B.42}$$

Proof. Equations (B.16), (B.17), (B.18) and (B.21) from Theorem B.7 can be applied in the exact same form. Equation (B.20) can be modified trivially to yield the analogous result.

Defining $\mathcal{F}_\tau = \{x_0, x_{1/2}, \ldots, x_\tau, x_{\tau+1/2}\}$, $\mathcal{F}'_\tau = \{x_0, x_{1/2}, \ldots, x_\tau\}$ as before, the analogous equation to Equation (B.19) is:

$$\begin{aligned}
\mathbb{E}\left[\|\tilde{F}_{\tau+1} - \tilde{F}_{\tau+1/2}\|_\star^2\right] &\leq \sum_{i=1}^n \mathbb{E}\left[\|\tilde{F}^{(i)}_{\tau+1} - \tilde{F}^{(i)}_{\tau+1/2}\|_\star^2\right] \leq \sum_{i=1}^n 2\left(\mathbb{E}\left[\|\tilde{F}^{(i)}_{\tau+1}\|_\star^2\right] + \mathbb{E}\left[\|\tilde{F}^{(i)}_{\tau+1/2}\|_\star^2\right]\right) \\
&= \sum_{i=1}^n \frac{2}{p_i^2}\left(\mathbb{E}\left[\mathbb{E}\left[\|M^{(i)}_{\tau+1/2}\hat{F}^{(i)}(x_{\tau+1/2})\|_\star^2|\mathcal{F}_\tau\right]\right] + \mathbb{E}\left[\mathbb{E}\left[\|M^{(i)}_{\tau}\hat{F}^{(i)}(x_{\tau})\|_\star^2|\mathcal{F}'_\tau\right]\right]\right) \\
&\leq \sum_{i=1}^n \frac{2}{p_i}\left(\mathbb{E}\left[\|\hat{F}^{(i)}(x_{\tau+1/2})\|_\star^2\right] + \mathbb{E}\left[\|\hat{F}^{(i)}(x_{\tau})\|_\star^2\right]\right) = \sum_{i=1}^n \frac{2\sum_{k=1}^n G_k}{npG_i} 2G_i^2 \leq \frac{4\left(\sum_{i=1}^n G_i\right)^2}{np}
\end{aligned} \tag{B.43}$$

Equation (B.22) can be rewritten as:

$$\begin{aligned}
\mathbb{E}\left[\|F(x_{\tau+1/2}) - \tilde{F}_{\tau+1}\|_\star^2\right] &\leq \sum_{i=1}^n \mathbb{E}\left[\|F^{(i)}(x_{\tau+1/2}) - \tilde{F}^{(i)}_{\tau+1}\|_\star^2\right] \\
&= \sum_{i=1}^n \mathbb{E}\left[\left\|F^{(i)}(x_{\tau+1/2}) - \frac{M^{(i)}_{\tau+1}}{p_i}\hat{F}^{(i)}(x_{\tau+1/2})\right\|_\star^2\right] \leq \sum_{i=1}^n 2\mathbb{E}\left[\left\|\left(I - \frac{M^{(i)}_{\tau+1}}{p_i}\right)\hat{F}^{(i)}(x_{\tau+1/2})\right\|_\star^2\right] \\
&\qquad + \sum_{i=1}^n 2\mathbb{E}\left[\left\|F^{(i)}(x_{\tau+1/2}) - \hat{F}^{(i)}(x_{\tau+1/2})\right\|_\star^2\right] \\
&\leq \sum_{i=1}^n 2\mathbb{E}\left[p_i\left\|\frac{p_i - 1}{p_i}\hat{F}^{(i)}(x_{\tau+1/2})\right\|_\star^2 + (1-p_i)\|\hat{F}^{(i)}(x_{\tau+1/2})\|_\star^2\right] + 2n\sigma^2 \\
&= \sum_{i=1}^n 2\left(1 - p_i + \frac{(1-p_i)^2}{p_i}\right)\mathbb{E}\left[\|\hat{F}^{(i)}(x_{\tau+1/2})\|_\star^2\right] + 2n\sigma^2 \\
&= \sum_{i=1}^n 2\frac{1-p_i}{p_i}G_i + 2n\sigma^2 \leq \frac{2\sum_{k=1}^n G_k}{npG_i}G_i^2 + 2n\sigma^2 \leq \frac{2(\sum_{i=1}^n G_i)^2}{np} + 2n\sigma^2
\end{aligned} \tag{B.44}$$

We conclude as in Theorem B.7. □

Remark B.14. By arithmetic mean-quadratic mean inequality,

$$\sum_{i=1}^n G_i^2 \geq \frac{(\sum_{i=1}^n G_i)^2}{n} \tag{B.45}$$

That implies that the bound with importance sampling is lower or equal than the original one.

Remark B.15. Corollary B.8 and Equations (B.29) and (B.30) hold when we substitute nG^2 by $(\sum_{i=1}^n G_i)^2/n$.

B.4 Mirror prox with variance reduced player stochasticity

Algorithm 5 Mirror prox with variance reduced player stochasticity

Initialization:
Choose stepsizes $\gamma_\tau > 0$ for $1 \leq \tau \leq t$.
Choose initial value $x_0 \in \mathbb{R}^{nd}$.
Set $G_0 = \hat{F}(x_0) \in \mathbb{R}^{nd}$

Main routine:
for $\tau = 0, \ldots, t$ **do**
    Sample the random matrices $M_\tau, M_{\tau+1/2} \in \mathbb{R}^{nd \times nd}$. Each diagonal block is the identity with probability p and 0 with probability $1-p$.
    Set $\tilde{F}_{\tau+1/2} = G_\tau + \frac{M_\tau}{p}(\hat{F}(\theta_\tau) - G_\tau)$
    Set $G_{\tau+1/2} = G_\tau + M_\tau(\hat{F}(\theta_\tau) - G_\tau)$
    Intermediate step: $\theta_{\tau+1/2} = P_{\theta_\tau}(\gamma_\tau \tilde{F}_{\tau+1/2})$.
    Set $\tilde{F}_{\tau+1} = G_{\tau+1/2} + \frac{M_{\tau+1/2}}{p}(\hat{F}(\theta_{\tau+1/2}) - G_{\tau+1/2})$
    Set $G_{\tau+1} = G_{\tau+1/2} + M_{\tau+1/2}(\hat{F}(\theta_{\tau+1/2}) - G_{\tau+1/2})$
    Extra-gradient step: $\theta_{\tau+1} = P_{\theta_\tau}(\gamma_\tau \tilde{F}_{\tau+1})$.
Return $\hat{\theta}_t$ where

$$\hat{x}_t = \left[\sum_{\tau=0}^{t} \gamma_\tau\right]^{-1} \sum_{\tau=0}^{t} \gamma_\tau \theta_{\tau+1/2}.$$

B.4.1 Proof

Lemma B.16. *The following holds:*

$$\mathbb{E}\left[\sup_{u\in Z} \sum_{\tau=0}^{t} \langle \gamma_\tau F(\theta_{\tau+1/2}), \theta_{\tau+1/2} - u\rangle\right] \leq \mathbb{E}\left[\sup_{u\in Z} 2D(u,\theta_0) - D(u,\theta_{t+1})\right]$$
$$+ \mathbb{E}\left[\sum_{\tau=0}^{t} 2\gamma_\tau^2 \|\tilde{F}_{\tau+1} - F(\theta_{\tau+1/2})\|_\star^2 + \frac{3\gamma_\tau^2}{2}\|F(\theta_\tau) - \tilde{F}_{\tau+1/2}\|_\star^2 + \frac{3\gamma_\tau^2}{2}\|F(\theta_{\tau+1/2}) - F(\theta_\tau)\|_\star^2\right]$$
$$+ \mathbb{E}\left[-\sum_{\tau=0}^{t} \frac{1}{2}\|\theta_{\tau+1/2} - \theta_\tau\|_2^2\right] \quad \text{(B.46)}$$

Proof. We rewrite Equation (B.18):

$$\langle \gamma_\tau \tilde{F}_{\tau+1}, \theta_{\tau+1/2} - u\rangle + D(u,\theta_{\tau+1}) - D(u,\theta_\tau) \leq \frac{\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1} - \tilde{F}_{\tau+1/2}\|_\star^2 - \frac{1}{2}\|\theta_{\tau+1/2} - \theta_\tau\|^2$$
$$\leq \frac{3\gamma_\tau^2}{2}\|\tilde{F}_{\tau+1} - F(\theta_{\tau+1/2})\|_\star^2 + \frac{3\gamma_\tau^2}{2}\|F(\theta_\tau) - \tilde{F}_{\tau+1/2}\|_\star^2$$
$$+ \frac{3\gamma_\tau^2}{2}\|F(\theta_{\tau+1/2}) - F(\theta_\tau)\|_\star^2 - \frac{1}{2}\|\theta_{\tau+1/2} - \theta_\tau\|^2 \quad \text{(B.47)}$$

We rewrite Equation (B.21). We have $\Delta_\tau = F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}$ and $y_\tau = P_{y_{\tau-1}}(\gamma_\tau)$ with $y_0 = x_0$.

$$\sum_{\tau=0}^{t}\langle\gamma_\tau\Delta_\tau, y_{\tau-1}-u\rangle \leq D(u,x_0) + \sum_{\tau=0}^{t}\frac{\gamma_\tau^2}{2}\|\Delta_\tau\|_\star^2$$
$$= D(u,x_0) + \sum_{\tau=0}^{t}\frac{\gamma_\tau^2}{2}\|F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}\|_\star^2. \quad \text{(B.48)}$$

Using Equation (B.48) and the analog to Equation (B.20), we get the desired inequality.

□

The main challenge is to bound $\mathbb{E}\left[\sum_{\tau=0}^{t}\gamma_\tau^2\|\tilde{F}_{\tau+1} - F(\theta_{\tau+1/2})\|_\star^2 + \gamma_\tau^2\|F(\theta_\tau) - \tilde{F}_{\tau+1/2}\|_\star^2\right]$, which we can also express as

$$\mathbb{E}\left[\sum_{\tau=0}^{t}\gamma_\tau^2\|\tilde{F}_{\tau+1} - F(\theta_{\tau+1/2})\|_\star^2 + \gamma_\tau^2\|F(\theta_\tau) - \tilde{F}_{\tau+1/2}\|_\star^2\right]$$
$$\overset{\text{Lemma B.6}}{\leq} \mathbb{E}\left[\sum_{\tau=0}^{t}\sum_{i=1}^{n}\gamma_\tau^2\|\tilde{F}_{\tau+1}^{(i)} - F^{(i)}(\theta_{\tau+1/2})\|_\star^2 + \gamma_\tau^2\|F^{(i)}(\theta_\tau) - \tilde{F}_{\tau+1/2}^{(i)}\|_\star^2\right] \quad \text{(B.49)}$$

Lemma B.17. *The following equalities hold:*

$$\mathbb{E}\left[\|\tilde{F}_{\tau+1}^{(i)} - F^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] = \frac{2(1-p)}{p}\mathbb{E}\left[\|G_{\tau+1/2}^{(i)} - \hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] + 2\sigma^2 \quad \text{(B.50)}$$

$$\mathbb{E}\left[\|F^{(i)}(\theta_\tau) - \tilde{F}_{\tau+1/2}^{(i)}\|_\star^2\right] = \frac{2(1-p)}{p}\mathbb{E}\left[\|G_{\tau}^{(i)} - \hat{F}^{(i)}(\theta_{\tau})\|_\star^2\right] + 2\sigma^2 \quad \text{(B.51)}$$

Proof. Using the conditional expectation with respect to the filtration up to w_τ,

$$\mathbb{E}\left[\|\tilde{F}_{\tau+1}^{(i)} - F^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] = 2\mathbb{E}\left[\left\|G_{\tau+1/2}^{(i)} + \frac{M_{\tau+1/2}^{(i)}}{p}(\hat{F}^{(i)}(\theta_{\tau+1/2}) - G_{\tau+1/2}^{(i)}) - \hat{F}^{(i)}(\theta_{\tau+1/2})\right\|_\star^2\right]$$
$$+2\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{\tau+1/2}) - F^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] = 2\mathbb{E}\left[\left\|\left(I - \frac{M_{\tau+1/2}^{(i)}}{p}\right)(G_{\tau+1/2}^{(i)} - \hat{F}^{(i)}(\theta_{\tau+1/2}))\right\|_\star^2\right] + 2\sigma^2$$
$$= 2\mathbb{E}\left[p\left\|\frac{p-1}{p}(G_{\tau+1/2}^{(i)} - \hat{F}^{(i)}(\theta_{\tau+1/2}))\right\|_\star^2 + (1-p)\|G_{\tau+1/2}^{(i)} - \hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] + 2\sigma^2$$
$$= 2\left(1 - p + \frac{(1-p)^2}{p}\right)\mathbb{E}\left[\|G_{\tau+1/2}^{(i)} - \hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] + 2\sigma^2$$
$$= \frac{2(1-p)}{p}\mathbb{E}\left[\|G_{\tau+1/2}^{(i)} - \hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] + 2\sigma^2 \quad \text{(B.52)}$$

The second equality is derived analogously. □

Let us define the change of variables $j = 2\tau$. Parametrized by j, the sequences that we are dealing with are $(M^{(i)}_{j/2})_{j\in\mathbb{N}}$, $(G^{(i)}_{j/2})_{j\in\mathbb{N}}$ and $(\theta_{j/2})_{j\in\mathbb{N}}$. In this scope i is a fixed integer between 1 and n.

Definition B.18. For a given j, let us define K_j as the random variable indicating the highest $k \in \mathbb{N}$ strictly lower than j such that $M^{(i)}_{k/2}$ is the identity (and $K_j = 0$ if there exists no such k).

In other words, K_j is the last step k before j at which the sequence $(G^{(i)}_{k/2})_{k\in\mathbb{N}}$ was updated with a new value $\hat{F}^{(i)}(\theta_{k/2})$. That is, $G_{j/2,i} = \hat{F}^{(i)}(\theta_{k_j/2})$.

Remark B.19. For a given j, $j - K_j$ is a random variable that has a geometric distribution with parameter p and support between 1 and j, i.e., for all k such that $j - 1 \geq k \geq 1$,

$$P(K_j = k) = p(1-p)^{j-1-k} \tag{B.53}$$

and $P(K_j = 0) = 1 - \sum_{k=1}^{j-1} P(K_j = k) = (1-p)^{j-1}$.

Lemma B.20. *The following inequality holds for any $j \in \mathbb{N}, p \in \mathbb{R}$ such that $p > 0$:*

$$\frac{(2\lceil (j+1)/2\rceil - j)(1-p)^{2\lceil (j+1)/2\rceil - j - 1}p + 2(1-p)^{2\lceil (j+1)/2\rceil - j}}{p^2} \leq \frac{2-p}{p^2} \tag{B.54}$$

Proof. For j even, we can write

$$(2\lceil (j+1)/2\rceil - j)(1-p)^{2\lceil (j+1)/2\rceil - j - 1}p + 2(1-p)^{2\lceil (j+1)/2\rceil - j} = 2(1-p)p + 2(1-p)^2 = 2(1-p) \tag{B.55}$$

For j odd,

$$(2\lceil (j+1)/2\rceil - j)(1-p)^{2\lceil (j+1)/2\rceil - j - 1}p + 2(1-p)^{2\lceil (j+1)/2\rceil - j} = p + 1 - p + 1 - p = 2 - p \tag{B.56}$$

Since $p > 0$, $2 - p \geq 2(1-p)$. □

Lemma B.21. *Let us define $h : \mathbb{R} \to \mathbb{R}$ as*

$$h(p) := \frac{2-p}{p^2} \tag{B.57}$$

Assume that $(\gamma_\tau)_{\tau\in\mathbb{N}}$ is non-increasing. Then, the following holds:

$$\sum_{\tau=0}^{t} \gamma_\tau^2 \mathbb{E}\left[\|G^{(i)}_\tau - \hat{F}^{(i)}(\theta_\tau)\|_\star^2\right] \leq \sum_{j=0}^{2t-1} h(p)\gamma^2_{\lfloor j/2\rfloor}\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_\star^2\right] \tag{B.58}$$

$$\sum_{\tau=0}^{t} \gamma_\tau^2 \mathbb{E}\left[\|G^{(i)}_{\tau+1/2} - \hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] \leq \sum_{j=0}^{2t-1} h(p)\gamma^2_{\lfloor j/2\rfloor}\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_\star^2\right] \tag{B.59}$$

Proof. We can write

$$\mathbb{E}\left[\|G_\tau^{(i)} - \hat{F}^{(i)}(\theta_\tau)\|_\star^2\right] = \mathbb{E}\left[\|G_{2\tau/2}^{(i)} - \hat{F}^{(i)}(\theta_{2\tau/2})\|_\star^2\right] = \mathbb{E}\left[\mathbb{E}\left[\|G_{2\tau/2}^{(i)} - \hat{F}^{(i)}(\theta_{2\tau/2})\|_\star^2 \middle| K_{2\tau}\right]\right]$$
$$= \sum_{k=0}^{2\tau-1} P(K_{2\tau}=k)\mathbb{E}\left[\|G_{2\tau/2}^{(i)} - \hat{F}^{(i)}(\theta_{2\tau/2})\|_\star^2 \middle| K_{2\tau}=k\right]$$
$$= \sum_{k=1}^{2\tau-1} p(1-p)^{2\tau-1-k}\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{k/2}) - \hat{F}^{(i)}(\theta_{2\tau/2})\|_\star^2\right] + (1-p)^{2\tau-1}\mathbb{E}\left[\|\hat{F}^{(i)}(x_0) - \hat{F}^{(i)}(\theta_{2\tau/2})\|_\star^2\right] \tag{B.60}$$

As seen in Equation (B.60), the point of conditioning with respect to the sigma-field generated by $K_{2\tau}$ is that we can write the expression for $G_{2\tau/2,i}$.

Now, using the rearrangement inequality,

$$\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{k/2}) - \hat{F}^{(i)}(\theta_{2\tau/2})\|_\star^2\right] = \mathbb{E}\left[\left\|\sum_{j=k}^{2\tau-1} \hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\right\|_\star^2\right]$$
$$\leq \sum_{j=k}^{2\tau-1}(2\tau-k)\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_\star^2\right] \tag{B.61}$$

Using Equations (B.60) and (B.61) we can now write

$$\sum_{\tau=0}^{t}\gamma_\tau^2\mathbb{E}\left[\|G_\tau^{(i)} - \hat{F}^{(i)}(\theta_\tau)\|_\star^2\right] = \sum_{\tau=0}^{t}\gamma_\tau^2\sum_{k=1}^{2\tau-1} p(1-p)^{2\tau-1-k}\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{k/2}) - \hat{F}^{(i)}(\theta_{2\tau/2})\|_\star^2\right]$$
$$+ \gamma_\tau^2(1-p)^{2\tau-1}\mathbb{E}\left[\|\hat{F}^{(i)}(x_0) - \hat{F}^{(i)}(\theta_{2\tau/2})\|_\star^2\right]$$
$$\leq \sum_{\tau=0}^{t}\gamma_\tau^2\sum_{k=1}^{2\tau-1} p(1-p)^{2\tau-1-k}\sum_{j=k}^{2\tau-1}(2\tau-k)\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_\star^2\right]$$
$$+ \gamma_\tau^2(1-p)^{2\tau-1}\sum_{j=0}^{2\tau-1} 2\tau\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_\star^2\right] \tag{B.62}$$

Given j between 0 and $2t-1$ the right hand side of Equation (B.62) contains the term $\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_2^2\right]$ multiplied by

$$\begin{aligned}
&\sum_{\tau=\lceil (j+1)/2\rceil}^{t} \gamma_\tau^2 \sum_{r=1}^{j}(2\tau - r)p(1-p)^{2\tau-1-r} + 2\tau(1-p)^{2\tau-1} \\
&\qquad \leq \gamma_{\lfloor j/2\rfloor}^2 \sum_{\tau=\lceil (j+1)/2\rceil}^{t} \sum_{r=1}^{j}(2\tau - r)p(1-p)^{2\tau-1-r} + 2\tau(1-p)^{2\tau-1} \\
&\qquad = \gamma_{\lfloor j/2\rfloor}^2 \sum_{\tau=\lceil (j+1)/2\rceil}^{t} p\sum_{r'=0}^{j-1}(1-p)^{2\tau-1-j+r'}(2\tau - j + r') + 2\tau(1-p)^{2\tau-1} \\
&\qquad \leq \gamma_{\lfloor j/2\rfloor}^2 \sum_{\tau=\lceil (j+1)/2\rceil}^{t} p\sum_{r'=2\tau-j}^{\infty}(1-p)^{r'-1}r' \\
&\qquad = \gamma_{\lfloor j/2\rfloor}^2 \sum_{\tau=\lceil (j+1)/2\rceil}^{t} p\frac{(2\tau - j)(1-p)^{2\tau-1-j}p + (1-p)^{2\tau-j}}{p^2} \\
&\qquad = \gamma_{\lfloor j/2\rfloor}^2 \sum_{\tau=\lceil (j+1)/2\rceil}^{t} \frac{(2\tau - j)(1-p)^{2\tau-1-j}p + (1-p)^{2\tau-j}}{p} \\
&\leq \gamma_{\lfloor j/2\rfloor}^2 \sum_{\tau=2\lceil (j+1)/2\rceil}^{\infty}(\tau - j)(1-p)^{\tau-1-j} + \frac{\gamma_{\lfloor j/2\rfloor}^2}{p}\sum_{\tau=2\lceil (j+1)/2\rceil}^{\infty}(1-p)^{\tau-j} \\
&\qquad = \gamma_{\lfloor j/2\rfloor}^2 \sum_{\tau=2\lceil (j+1)/2\rceil-j}^{\infty}\tau(1-p)^{\tau-1} + \frac{\gamma_{\lfloor j/2\rfloor}^2}{p}\sum_{\tau=2\lceil (j+1)/2\rceil-j}^{\infty}(1-p)^{\tau} \\
&\qquad = \gamma_{\lfloor j/2\rfloor}^2 \frac{(2\lceil (j+1)/2\rceil - j)(1-p)^{2\lceil (j+1)/2\rceil-j-1}p + 2(1-p)^{2\lceil (j+1)/2\rceil-j}}{p^2}
\end{aligned} \tag{B.63}$$

We have used twice that for $|\alpha| < 1$

$$\sum_{s=q}^{\infty}\alpha^{s-1}s = \left(\sum_{s=q}^{\infty}\alpha^s\right)' = \left(\frac{\alpha^q}{1-\alpha}\right)' = \frac{q\alpha^{q-1}(1-\alpha) + \alpha^q}{(1-\alpha)^2} \tag{B.64}$$

By Lemma B.20 we have

$$\frac{(2\lceil (j+1)/2\rceil - j)(1-p)^{2\lceil (j+1)/2\rceil-j-1}p + 2(1-p)^{2\lceil (j+1)/2\rceil-j}}{p^2} \leq h(p) \tag{B.65}$$

Hence, from Equation (B.62) we get

$$\sum_{\tau=0}^{t}\gamma_\tau^2\mathbb{E}\left[\|G_\tau^{(i)} - \hat{F}^{(i)}(\theta_\tau)\|_\star^2\right] \leq \sum_{j=0}^{2t-1}\gamma_{\lfloor j/2\rfloor}^2 h(p)\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_\star^2\right] \tag{B.66}$$

Analogously to Equation (B.60):

$$\begin{aligned}\mathbb{E}\left[\|G^{(i)}_{\tau+1/2} - \hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] &= \mathbb{E}\left[\|G^{(i)}_{(2\tau+1)/2} - \hat{F}^{(i)}(\theta_{(2\tau+1)/2})\|_\star^2\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[\|G^{(i)}_{(2\tau+1)/2} - \hat{F}^{(i)}(\theta_{(2\tau+1)/2})\|_\star^2 \middle| K_{2\tau+1}\right]\right] \\ &= \sum_{k=0}^{2\tau} P(K_{2\tau+1}=k)\mathbb{E}\left[\|G^{(i)}_{(2\tau+1)/2} - \hat{F}^{(i)}(\theta_{(2\tau+1)/2})\|_\star^2 \middle| K_{2\tau+1}=k\right] \\ = \sum_{k=1}^{2\tau} p(1-p)^{2\tau-k}\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{k/2}) - \hat{F}^{(i)}(\theta_{(2\tau+1)/2})\|_\star^2\right] &+ (1-p)^{2\tau}\mathbb{E}\left[\|\hat{F}^{(i)}(x_0) - \hat{F}^{(i)}(\theta_{(2\tau+1)/2})\|_\star^2\right]\end{aligned} \tag{B.67}$$

Using the same reasoning we get an inequality that is analogous to Equation (B.58):

$$\sum_{\tau=0}^{t}\gamma_\tau^2\mathbb{E}\left[\|G^{(i)}_{\tau+1/2} - \hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] \leq \sum_{j=0}^{2t}\gamma^2_{\lfloor j/2\rfloor}h(p)\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_\star^2\right] \tag{B.68}$$

□

Lemma B.22. *Assume that for all i between 1 and n, the gradients $\nabla_i \ell_i$ are L-Lipschitz. Assume that for all τ between 0 and t, $\gamma_\tau \leq \gamma$. Let*

$$\chi(p,\gamma) = 1 - 36\frac{1-p}{p}nh(p)L^2\gamma^2 \tag{B.69}$$

If γ is small enough that $\chi(p,\gamma)$ is positive, then

$$\begin{aligned}&\mathbb{E}\left[\sum_{\tau=0}^{t}\gamma_\tau^2\|\tilde{F}_{\tau+1} - F(\theta_{\tau+1/2})\|_\star^2 + \gamma_\tau^2\|F(\theta_\tau) - \tilde{F}_{\tau+1/2}\|_\star^2\right] \\ &\qquad \leq 16n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2 + \frac{1-p}{p\chi(p,\gamma)}(12L^2 + 36L^4\gamma^2)nh(p)\sum_{\tau=0}^{t}\gamma_\tau^2\mathbb{E}\left[\|\theta_\tau - \theta_{\tau+1/2}\|_\star^2\right]\end{aligned} \tag{B.70}$$

Proof. We first want to bound the terms $\mathbb{E}\left[\|F^{(i)}(\theta_{j/2}) - F^{(i)}(\theta_{(j+1)/2})\|_2^2\right]$. When j is even we can make the change of variables $j/2 = \tau$ (just for simplicity in the notation) and use smoothness. We get

$$\begin{aligned}\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_\star^2\right] &= \mathbb{E}\left[\|\hat{F}^{(i)}(\theta_\tau) - \hat{F}^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] \\ &\leq 3\mathbb{E}\left[\|F^{(i)}(\theta_\tau) - F^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] + 3\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_\tau) - F^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] \\ &\quad + 3\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{\tau+1/2}) - F^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] \leq 3L^2\mathbb{E}\left[\|\theta_\tau - \theta_{\tau+1/2}\|_\star^2\right] + 6\sigma^2\end{aligned} \tag{B.71}$$

When j is odd, we can write $j/2 = \tau + 1/2$. We use smoothness and the fact that the prox-mapping is 1-Lipschitz (Lemma B.1):

$$\begin{aligned}
\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{j/2}) - \hat{F}^{(i)}(\theta_{(j+1)/2})\|_\star^2\right] &= \mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{\tau+1/2}) - \hat{F}^{(i)}(\theta_{\tau+1})\|_\star^2\right] \\
&\leq 3\mathbb{E}\left[\|F^{(i)}(\theta_{\tau+1/2}) - F^{(i)}(\theta_{\tau+1})\|_\star^2\right] + 3\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{\tau+1/2}) - F^{(i)}(\theta_{\tau+1/2})\|_\star^2\right] \\
&\quad + 3\mathbb{E}\left[\|\hat{F}^{(i)}(\theta_{\tau+1}) - F^{(i)}(\theta_{\tau+1})\|_\star^2\right] \leq 3L^2\mathbb{E}\left[\|\theta_{\tau+1/2} - \theta_{\tau+1}\|_\star^2\right] + 6\sigma^2 \\
&= 3L^2\mathbb{E}\left[\|P_{\theta_\tau}(\gamma_\tau \tilde{F}_{\tau+1/2}) - P_{\theta_\tau}(\gamma_\tau \tilde{F}_{\tau+1})\|_\star^2\right] + 6\sigma^2 \leq 3L^2\gamma_\tau^2\mathbb{E}\left[\|\tilde{F}_{\tau+1/2} - \tilde{F}_{\tau+1}\|_\star^2\right] + 6\sigma^2 \\
&\leq 9L^2\gamma_\tau^2\left(\mathbb{E}\left[\|\tilde{F}_{\tau+1/2} - F(\theta_\tau)\|_\star^2\right] + \mathbb{E}\left[\|F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}\|_\star^2\right] + \mathbb{E}\left[\|F(\theta_\tau) - F(\theta_{\tau+1/2})\|_\star^2\right]\right) \\
&\quad + 6\sigma^2 \qquad \text{(B.72)}
\end{aligned}$$

Hence, from Equations (B.49), (B.52), (B.51), (B.71) and (B.72):

$$\begin{aligned}
&\mathbb{E}\left[\sum_{\tau=0}^{t}\gamma_\tau^2\|\tilde{F}_{\tau+1} - F(\theta_{\tau+1/2})\|_\star^2 + \gamma_\tau^2\|F(\theta_\tau) - \tilde{F}_{\tau+1/2}\|_\star^2\right] \\
&\quad\leq 4n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2 + \frac{2(1-p)}{p}\sum_{i=1}^{n}\sum_{j=0}^{2t}2\gamma_{\lfloor j/2\rfloor}^2 h(p)\mathbb{E}\left[\|F_i(\theta_{j/2}) - F_i(\theta_{(j+1)/2})\|_\star^2\right] \\
&\quad= 4n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2 + \frac{2(1-p)}{p}\sum_{i=1}^{n}\sum_{j=0,\ j\text{ even}}^{2t}2\gamma_{\lfloor j/2\rfloor}^2 h(p)\mathbb{E}\left[\|F_i(\theta_{j/2}) - F_i(\theta_{(j+1)/2})\|_\star^2\right] \\
&\qquad+ \frac{2(1-p)}{p}\sum_{i=1}^{n}\sum_{j=0,\ j\text{ odd}}^{2t}2\gamma_{\lfloor j/2\rfloor}^2 h(p)\mathbb{E}\left[\|F_i(\theta_{j/2}) - F_i(\theta_{(j+1)/2})\|_\star^2\right] \\
&\quad= 4n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2 + \frac{2(1-p)}{p}\sum_{i=1}^{n}\sum_{\tau=0}^{t}2\gamma_\tau^2 h(p)\mathbb{E}\left[\|F_i(\theta_\tau) - F_i(\theta_{\tau+1/2})\|_\star^2\right] \\
&\qquad+ \frac{2(1-p)}{p}\sum_{i=1}^{n}\sum_{\tau=0}^{t}2\gamma_\tau^2 h(p)\mathbb{E}\left[\|F_i(\theta_{\tau+1/2}) - F_i(\theta_{\tau+1})\|_\star^2\right] \\
&\quad\leq 16n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2 + \frac{1-p}{p}\sum_{\tau=0}^{t}12n\gamma_\tau^2 h(p)L^2\mathbb{E}\left[\|\theta_\tau - \theta_{\tau+1/2}\|_\star^2\right] \\
&\qquad+ \frac{1-p}{p}\sum_{\tau=0}^{t}36nh(p)L^2\gamma_\tau^4\left(\mathbb{E}\left[\|\tilde{F}_{\tau+1/2} - F(\theta_\tau)\|_\star^2\right] + \mathbb{E}\left[\|F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}\|_\star^2\right]\right) \\
&\qquad\quad+ \frac{1-p}{p}\sum_{\tau=0}^{t}36nh(p)L^4\gamma_\tau^4\mathbb{E}\left[\|\theta_\tau - \theta_{\tau+1/2}\|_\star^2\right] \\
&\quad\leq 16n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2 + \frac{1-p}{p\chi(p,\gamma)}(12L^2 + 36L^4\gamma^2)nh(p)\sum_{\tau=0}^{t}\gamma_\tau^2\mathbb{E}\left[\|\theta_\tau - \theta_{\tau+1/2}\|_\star^2\right] \\
&\qquad+ 36\frac{1-p}{p}nh(p)L^2\gamma^2\sum_{\tau=0}^{t}\gamma_\tau^2\left(\mathbb{E}\left[\|\tilde{F}_{\tau+1/2} - F(\theta_\tau)\|_\star^2\right] + \mathbb{E}\left[\|F(\theta_{\tau+1/2}) - \tilde{F}_{\tau+1}\|_\star^2\right]\right) \qquad \text{(B.73)}
\end{aligned}$$

Rearranging and using $\chi(p,\gamma) > 0$ yields the desired result. □

Theorem B.23. *Assume that for all i between 1 and n, the gradients $\nabla_i \ell_i$ are L-Lipschitz. Let $(\hat{\theta}_t)_{t\in\mathbb{N}}$ be defined as in Algorithm 5. Choose $(\gamma_t)_{t\in\mathbb{N}}$ such that $\gamma_t \leq \gamma$, with γ defined as*

$$\gamma := \min\left\{\frac{p^{3/2}}{\sqrt{(1-p)(2-p)}}\frac{1}{12L\sqrt{n}}, \frac{1}{L}\sqrt{\frac{7}{27n+12}}\right\} \tag{B.74}$$

Then,

$$\mathrm{Err}_N(\hat{\theta}_t) \leq \left(\sum_{\tau=0}^{t}\gamma_\tau\right)^{-1}\left(2\Omega + 32n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2\right) \tag{B.75}$$

Proof. It is easy to see that

$$\gamma \leq \frac{p^{3/2}}{\sqrt{(1-p)(2-p)}}\frac{1}{12L\sqrt{n}} \iff \chi(p,\gamma) \geq 3/4 > 0 \tag{B.76}$$

Hence, the assumptions of Lemma B.22 are fulfilled. Starting from the result in Lemma B.16 and using Lemma B.22,

$$\begin{aligned}
\mathbb{E}\left[\sup_{u\in Z}\sum_{\tau=0}^{t}\langle\gamma_\tau F(\theta_{\tau+1/2}), \theta_{\tau+1/2}-u\rangle\right] &\leq \mathbb{E}\left[\sup_{u\in Z} 2D(u,\theta_0) - D(u,\theta_t)\right] + 32n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2 \\
&\quad + 2\frac{1-p}{p\chi(p,\gamma)}(12L^2+36L^4\gamma^2)nh(p)\sum_{\tau=0}^{t}\gamma_\tau^2\mathbb{E}\left[\|\theta_\tau-\theta_{\tau+1/2}\|_\star^2\right] \\
&\quad + \frac{3nL^2}{2}\sum_{\tau=0}^{t}\gamma_\tau^2\mathbb{E}\left[\|\theta_\tau-\theta_{\tau+1/2}\|_\star^2\right] - \frac{1}{2}\sum_{\tau=0}^{t}\mathbb{E}\left[\|\theta_\tau-\theta_{\tau+1/2}\|_\star^2\right] \\
\leq 2\Omega + 32n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2 &+ \left((24L^2+72L^4\gamma^2)nh(p)\gamma^2\frac{1-p}{p\chi(p,\gamma)} + \frac{3n\gamma^2L^2}{2} - \frac{1}{2}\right)\sum_{\tau=0}^{t}\mathbb{E}\left[\|\theta_\tau-\theta_{\tau+1/2}\|_\star^2\right]
\end{aligned} \tag{B.77}$$

Recalling the definition of $h(p)$ in Equation (B.57), the conditions $\chi(p,\gamma) \geq 3/4$ and

$$\gamma \leq \frac{1}{L}\sqrt{\frac{7}{27n+12}} \tag{B.78}$$

imply

$$(24L^2+72L^4\gamma^2)nh(p)\gamma^2\frac{1-p}{p\chi(p,\gamma)} + \frac{3n\gamma^2L^2}{2} - \frac{1}{2} \leq 0 \tag{B.79}$$

We show this development because it is not entirely trivial:

$$\begin{aligned}
&(24L^2 + 72L^4\gamma^2)n\frac{2-p}{p^2}\gamma^2\frac{1-p}{p\chi(p,\gamma)} + \frac{3n\gamma^2L^2}{2} - \frac{1}{2} \\
&\overset{\chi\geq 3/4}{\leq} (24L^2+72L^4\gamma^2)n\frac{2-p}{p^2}\gamma^2\frac{4(1-p)}{3p} + \frac{3n\gamma^2L^2}{2} - \frac{1}{2} = \frac{24+72L^2\gamma^2}{27}(1-\chi(p,\gamma)) + \frac{3n\gamma^2L^2}{2} - \frac{1}{2} \\
&\leq \frac{2+6L^2\gamma^2}{9} + \frac{3n\gamma^2L^2}{2} - \frac{1}{2} = \gamma^2\frac{(9n+4)L^2}{6} - \frac{7}{18} \qquad \text{(B.80)}
\end{aligned}$$

Using Equation (B.79) on (B.77) yields

$$\mathbb{E}\left[\sup_{u\in Z}\sum_{\tau=1}^{t}\langle\gamma_\tau F(\theta_{\tau+1/2}), \theta_{\tau+1/2} - u\rangle\right] \leq 2\Omega + 32n\sigma^2\sum_{\tau=0}^{t}\gamma_\tau^2 \qquad \text{(B.81)}$$

We conclude by Lemma B.4. □

Corollary B.24. *Set γ as in Equation (B.74). For all τ between 1 and t, let*

$$\gamma_\tau = \min\left\{\gamma, \frac{1}{4}\sqrt{\frac{\Omega}{n\sigma^2 t}}\right\} \qquad \text{(B.82)}$$

Then,

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_t)\right] \leq \max\left\{\frac{4\Omega}{\gamma t}, 16\sqrt{\frac{\Omega n\sigma^2}{t}}\right\} \qquad \text{(B.83)}$$

Proof. Same proof as Corollary B.11. □

B.4.2 Bound in terms of the number of gradient computations

As in Subsections B.2.2 and B.2.4, we write $p = k/n$, $t(c) = c/(2k)$ for Equation (B.83):

$$\begin{aligned}
\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_{t(c)})\right] &\leq \max\left\{\frac{4\Omega}{\frac{\left(\frac{k}{n}\right)^{3/2}}{\sqrt{(1-\frac{k}{n})(2-\frac{k}{n})}}\frac{1}{12L\sqrt{n}}\frac{c}{2k}}, \frac{4\Omega}{\frac{1}{L}\sqrt{\frac{7}{27n+12}}\frac{c}{2k}}, 8\sqrt{\frac{2\Omega nk\sigma^2}{c}}\right\} \\
&\leq \max\left\{\frac{96\sqrt{2}\Omega Ln^2}{\sqrt{k}c}, 8\Omega kL\sqrt{\frac{27n+12}{7}}\frac{1}{c}, 16\sqrt{\frac{2\Omega nk\sigma^2}{c}}\right\} \qquad \text{(B.84)}
\end{aligned}$$

For comparison, starting from Equation 26 from [11], we get (in our notation)

$$\mathbb{E}\left[\mathrm{Err}_N(\hat{\theta}_{t(c)})\right] \leq \max\left\{7\frac{\Omega Ln^{3/2}}{c}, 14n\sqrt{\frac{2\Omega\sigma^2}{3c}}\right\} \qquad \text{(B.85)}$$

Appendix C

Supplementary Material: Structured Planning

C.1 State-Attribute Regressor and Parametrization

In the case where f is not given by the user, on can train an estimator $\hat{f}$ from labeled pairs $\{s_i, \rho_i\}_{i \leq I} \in (\mathcal{S} \times \mathbb{R})^I$, with a neural network trained with a mean-squared loss that reflects the geometry of each target attribute coordinate. If $\mathbb{R} = G_1 \times \dots G_K$, $\rho = (\rho[1], \dots, \rho[K])$, and $y = (y[1], \dots, y[K])$ is the output of the neural net regressor, we consider the following metric on each G_k:

- If $G_k = \mathbb{N}$, then $y[k] \in \mathbb{R}$, $\ell_k(y[k], \rho[k]) = |y[k] - \rho[k]|^2$ and $\hat{\rho[k]} = \lceil y[k] \rceil$.
- If $G_k = \mathbb{R}$, then $y[k] \in \mathbb{R}$, $\ell_k(y[k], \rho[k]) = |y[k] - \rho[k]|^2$ and $\hat{\rho[k]} = y[k]$.
- If $G_k = \mathbb{Z}/(q\mathbb{Z})$, then $y[k] \in S^1$, $y[k] = \frac{u}{\|u\|} = (\sin\theta, \cos\theta)$, $\ell_k(y[k], \rho[k]) = 1 - \langle y[k], e^{2\pi i \rho[k]/q} \rangle$ and $\hat{\rho[k]} = \lceil \theta q / 2\pi \rceil$.

The loss aggregated through all attribute coordinates becomes $\ell(y, \rho) = \sum_{k \leq K} \ell_k(y[k], \rho[k])$.

C.2 Further Experimental Setup

C.2.1 Training specifications of the policies

For all the policies of our two models and all baselines we have used networks with two fully-connected hidden layers. Each hidden layer has 128 units. The batch size was of 5000 steps for the policies.

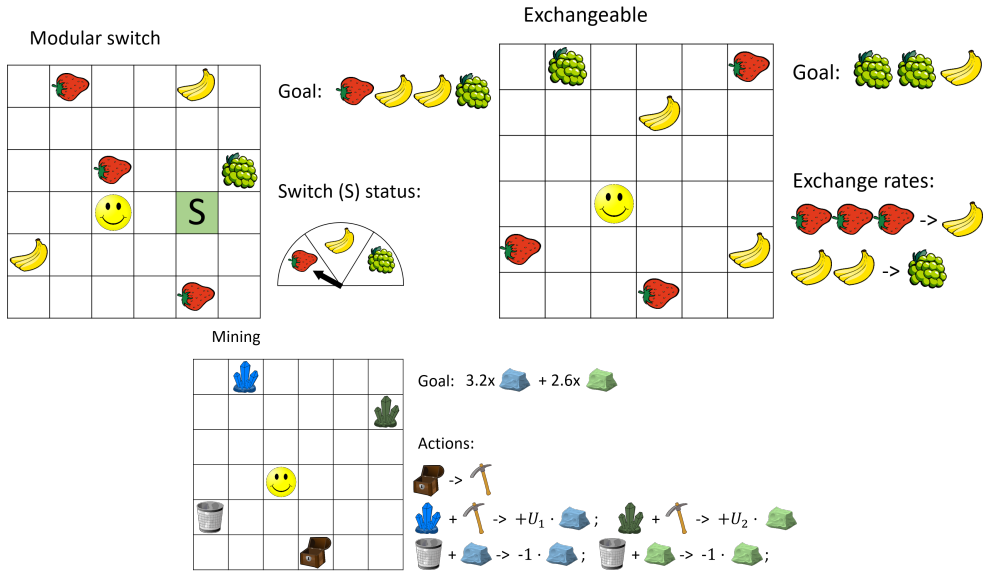

FIGURE C.1: Diagrams of the grid environments. *left:* Modular, *right:* Exchangeable, *center:* Mining

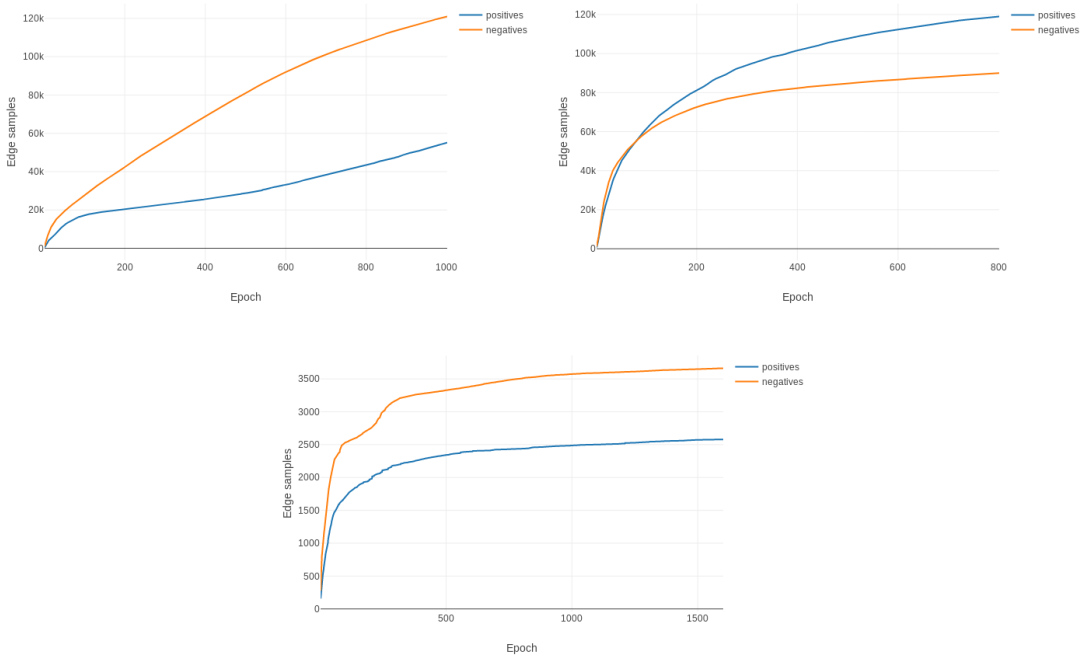

FIGURE C.2: Rate at which positive and negative examples are being generated by π_{expl} and π_{exec} on the three environments (*left:* Modular, *right:* Exchangeable, *center:* Mining)

The execution policies in both models have been trained using Generalized Advantage Estimation [60]. The expression for the gradient corresponding to a batch is:

$$\hat{g} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{\infty} \hat{A}_{n,t}^{GAE(\gamma,\tau)} \nabla_\theta \log \pi_\theta(a_t^n | s_t^n)$$

where

$$\hat{A}_{n,t}^{GAE(\gamma,\tau)} = \sum_{l=0}^{\infty} (\gamma\tau)^l \delta_{t+l}^V$$

with $\delta_{t+l}^V = r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l})$

We have used $\gamma = 0.9$ and $\tau = 1$.

For the exploration policy we have used the regular policy gradient method with discounted rewards.

C.2.2 Modular Switches

For the Simple Structured Attribute Model, the simple attribute differences we have used are: $(-1, 0, 0, 1, 0, 0, 0)$ / picking object 1, $(0, -1, 0, 0, 1, 0, 0)$ / picking object 2, $(0, 0, -1, 0, 0, 1, 0)$ / picking object 3, $(0, 0, 0, 0, 0, 0, 1)$ / pressing the switch. For the path inference in the Simplified Model, we have used that all the components of valid attributes are positive and that picking object i is only possible if the last component modulo 3 is i.

We have trained the exploration policy of the Structured Attribute Model with $\alpha_1 = 1$ and $\alpha_2 = 0$.

C.2.3 Exchangeable Attributes

In this experiment we have made the rewards of the exploration continuous in time, in the sense that on each step we give reward not just for the transition that finishes the episode, but also for all the transitions that come after that episode until the end of the game. This way we are encouraging the explorer to look for trajectories that lead to late unseen attributes. This didn't work on the other experiments, because it stimulates the policy to do as many transitions as possible, and it got stuck in places where one could execute a number of transitions in a row (mostly the switch, but also the hammer store, the dump, etc).

For the Simple Structured Attribute Model, the simple attribute differences we have used are: $(-1, 0, 0, 1, 0, 0)$ / picking object 1, $(0, -1, 0, 0, 1, 0)$ / picking object 2, $(0, 0, -1, 0, 0, 1)$ / picking object 3, $(0, 0, 0, -3, 1, 0)$ / trading object 1 for object 2, $(0, 0, 0, 0, -3, 2)$ / trading object 2 for

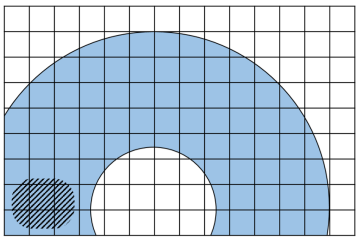

FIGURE C.3: Subset $\mathcal{R}'$ of the attributes allowed in the constrained attributes game (in blue), shown in the attribute space (only the first two components of the attributes are depicted, as the constraint we impose does not affect the third component). The black strips show the area where the game starts.

object 3, $(0, 0, 0, 1, 0, -2)$ / trading object 3 for object 1. For the path inference in the Simplified Model, we have only used that all the components of valid attributes are positive.

We have trained the exploration policy of the Structured Attribute Model with $\alpha_1 = 0$ and $\alpha_2 = 1$.

C.2.4 Constrained Attributes

In this game the transitions are only admissible as long as $\rho_{i+1} \in \mathcal{S}$. In our experiments we have defined this subset as $\mathcal{S} = \{(a_1, a_2, a_3) \in \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{N} \mid 2.5 \leq d((a_1, a_2), (6, 1)) \leq 7\}$. This set is illustrated in Figure C.3 as the blue zone. In the figure, the black strips show the area where the agent starts the game.

For the Simple Structured Attribute Model, the simple attribute differences we have used are: $(3, 0, -1)$ and $(2, 0, -1)$ / mining mineral 1, $(0, 3, -1)$ and $(0, 2, -1)$ / mining mineral 2, $(0, 0, 1)$ / getting a hammer, $(-1, 0, 0)$ / throwing mineral 1, $(0, -1, 0)$ / throwing mineral 2. Even though the first two components of the attributes are continuous, we can use integer simple attribute differences because we round attributes to determine if goals have been reached. For the path inference in the Simple Model, we have only used that in valid attributes all the components are positive.

We have trained the exploration policy of the Structured Attribute Model with $\alpha_1 = 0$ and $\alpha_2 = 1$.

C.2.5 Starcraft

For the Simple Structured Attribute Model, the simple attribute differences we have used are: $(8, 0, 0, 0, 0, 0)$ / mining ore, $(-50, 1, 0, 0, 0, 0)$ / building SCV, $(-150, 0, 0, 0, 0, 1)$ / building Barracks (not finished yet), $(0, 0, 1, 0, 0, -1)$ / Barracks are finished, $(-50, 0, 0, 1, 0, 0)$ / Building Marine, $(-100, 0, 0, 0, 1, 0)$ / Building Supply Depot. For the path inference in the Simplified

Model, we have used that in valid attributes all the components are positive and that Marines can only be built if at least one Barracks have been finished.

We have trained the exploration policy of the Structured Attribute Model with $\alpha_1 = 0$ and $\alpha_2 = 1$.

Bibliography

[1] Dov Monderer and Lloyd S. Shapley. Potential games. *Games and Economic Behavior*, 14: 124–143, 1996.

[2] David Balduzzi, Sébastien Racanière, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. In *International Conference on Machine Learning (ICML)*, 2018.

[3] Yang Cai, Ozan Candogan, Constantinos Daskalakis, and Christos Papadimitriou. Zero-sum polymatrix games: A generalization of minmax. *Mathematics of Operations Research*, 51(2):648–655, 2016.

[4] Arkadi Nemirovski, Shmuel Onn, and Uriel G. Rothblum. Accuracy certificates for computational problems with convex structure. *Mathematics of Operations Research*, 35(1): 52–78, 2010.

[5] Alistair Letcher, Jakob Foerster, David Balduzzi, Tim Rocktäschel, and Shimon Whiteson. Stable opponent shaping in differentiable games. In *International Conference on Learning Representations (ICLR)*, 2019.

[6] Chongjie Zhang and Victor Lesser. Multi-agent learning with policy prediction. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[7] Jakob Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018.

[8] GM Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

[9] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

[10] Arkadi Nemirovski. Prox-method with rate of convergence o (1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

[11] Anatoli Juditsky, Arkadi Nemirovski, and Claire Tauvel. Solving variational inequalities with stochastic mirror-prox algorithm. *Stochastic Systems*, 1(1):17–58, 2011.

[12] P. Lancaster. On eigenvalues of matrices dependent on a parameter. *Numerische Mathematik*, 6():377–387, 1964.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[14] Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Advances in neural information processing systems*, pages 5690–5701, 2017.

[15] Greg Wayne and LF Abbott. Hierarchical control using networks trained with higher-level forward models. *Neural computation*, 26(10):2163–2193, 2014.

[16] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3540–3549. JMLR. org, 2017.

[17] Lucian Bu, Robert Babu, Bart De Schutter, et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[18] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[19] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech M Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. *arXiv preprint arXiv:1901.08106*, 2019.

[20] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[21] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[22] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[23] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

[24] Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. *arXiv preprint arXiv:1802.10551*, 2018.

[25] Arkadi Nemirovski, Shmuel Onn, and Uriel G Rothblum. Accuracy certificates for computational problems with convex structure. *Mathematics of Operations Research*, 35(1): 52–78, 2010.

[26] Patrick T Harker and Jong-Shi Pang. Finite-dimensional variational inequality and nonlinear complementarity problems: a survey of theory, algorithms and applications. *Mathematical programming*, 48(1-3):161–220, 1990.

[27] Thomas L Magnanti and Georgia Perakis. Averaging schemes for variational inequalities and systems of equations. *Mathematics of operations research*, 22(3):568–587, 1997.

[28] Angelia Nedić and Asuman Ozdaglar. Subgradient methods for saddle-point problems. *Journal of optimization theory and applications*, 142(1):205–228, 2009.

[29] Brendan Jou and Shih-Fu Chang. Deep cross residual learning for multitask visual recognition. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 998–1007. ACM, 2016.

[30] Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J Pal. Learning general purpose distributed sentence representations via large scale multi-task learning. *arXiv preprint arXiv:1804.00079*, 2018.

[31] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[32] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016.

[33] George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. pearson education, 2005.

[34] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[35] J Ben Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, 3(33), 1964.

[36] Francisco Facchinei and Christian Kanzow. Generalized nash equilibrium problems. *4or*, 5 (3):173–210, 2007.

[37] Yurii Nesterov. Dual extrapolation and its applications to solving variational inequalities and related problems. *Mathematical Programming*, 109(2-3):319–344, 2007.

[38] Farzad Yousefian, Angelia Nedić, and Uday V Shanbhag. Optimal robust smoothing extragradient algorithms for stochastic variational inequality problems. In *53rd IEEE Conference on Decision and Control*, pages 5831–5836. IEEE, 2014.

[39] Francis Bach and Kfir Levy. A universal algorithm for variational inequalities adaptive to smoothness and noise. *arXiv:1902.01637*, 2019.

[40] Farzad Yousefian, Angelia Nedić, and Uday V Shanbhag. On stochastic mirror-prox algorithms for stochastic cartesian variational inequalities: Randomized block coordinate and optimal averaging schemes. *Set-Valued and Variational Analysis*, 26(4):789–819, 2018.

[41] Balamurugan Palaniappan and Francis Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, pages 1416–1424, 2016.

[42] AN Iusem, Alejandro Jofré, Roberto I Oliveira, and Philip Thompson. Extragradient method with variance reduction for stochastic variational inequalities. *SIAM Journal on Optimization*, 27(2):686–724, 2017.

[43] Tatjana Chavdarova, Gauthier Gidel, Francois Fleuret, Chuan-Sheng Foo, and Simon Lacoste-Julien. Reducing noise in gan training with variance reduced extragradient. *arXiv:1904.08598*, 2019.

[44] Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the 17th international joint conference on Artificial intelligence*, volume 2, pages 1021–1026, 2001.

[45] Vincent Conitzer and Tuomas Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, 2007.

[46] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training gans with optimism. *arXiv preprint arXiv:1711.00141*, 2017.

[47] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of gans. In *Advances in Neural Information Processing Systems*, pages 1825–1835, 2017.

[48] Panayotis Mertikopoulos, Houssam Zenati, Bruno Lecouat, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Mirror descent in saddle-point problems: Going the extra (gradient) mile. *arXiv preprint arXiv:1807.02629*, 2018.

[49] Vaishnavh Nagarajan and J Zico Kolter. Gradient descent gan optimization is locally stable. In *Advances in Neural Information Processing Systems*, pages 5585–5595, 2017.

[50] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[51] Eric V Mazumdar, Michael I Jordan, and S Shankar Sastry. On finding local nash equilibria (and only local nash equilibria) in zero-sum games. *arXiv preprint arXiv:1901.00838*, 2019.

[52] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

[53] Amy Zhang, Adam Lerer, Sainbayar Sukhbaatar, Rob Fergus, and Arthur Szlam. Composable planning with attributes. In *International Conference on Machine Learning (ICML)*, 2018.

[54] Natalia Hernandez-Gardiol and Leslie Pack Kaelbling. Envelope-based planning in relational mdps. In *NIPS*, pages 783–790. MIT Press, 2003.

[55] M. van Otterlo. *A Survey of Reinforcement Learning in Relational Domains.* Number 31 in TR-CTIT-05, ISSN 1381-3625. Centre for Telematics and Information Technology University of Twente, 2005.

[56] Carlos Diuk, Andre Cohen, and Michael L. Littman. An object-oriented representation for efficient reinforcement learning. In *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 240–247. ACM, 2008.

[57] David Abel, D. Ellis Hershkowitz, Gabriel Barth-Maron, Stephen Brawner, Kevin O'Farrell, James MacGlashan, and Stefanie Tellex. Goal-based action priors. In *ICAPS*, pages 306–314. AAAI Press, 2015.

[58] Sainbayar Sukhbaatar, Arthur Szlam, Gabriel Synnaeve, Soumith Chintala, and Rob Fergus. Mazebase: A sandbox for learning from games. *arXiv preprint arXiv:1511.07401*, 2015.

[59] Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothée Lacroix, Zeming Lin, Florian Richoux, and Nicolas Usunier. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*, 2016.

[60] John Schulman, Phillipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Machine Learning (ICML)*, 2015.